

# PRIME

Preliminary Documentation Release

REV. 14, Ø

NOV. 1977

OBSOLETE

**THE NEW  
USER'S GUIDE  
TO EDITOR  
AND RUNOFF  
PDR3104**

21 Ø. PAGES.

00APR82

## THE NEW USER'S GUIDE TO EDITOR & RUNOFF

This guide has been written primarily for people who have never used computers, text-editors, or text-processors before.

Programmers and other experienced users may turn directly to the command summaries in Appendix A. Complete information on all EDITOR and RUNOFF commands can be found in Sections 8 and 9.

This guide also replaces all previous documentation on both EDITOR and RUNOFF. (Note: The current versions of Prime's COBOL and FORTRAN manuals contain summaries of the relevant EDITOR commands.)

Please send all comments and suggestions to:

Daniel P. Dern, Technical Editor  
Technical Publications Department  
Prime Computer, Inc.  
145 Pennsylvania Avenue  
Framingham, Mass. 01701

### Acknowledgements

Thanks to everyone who helped: the EDITOR and RUNOFF team members, other participating customers and Prime employees, and, for the artwork on page 1-4, Brian Bradley.

#### PRIME DOCUMENTATION TYPES

- IDR Initial Documentation Release: provides usable, accurate advanced information without regard to style and format.
- PDR Preliminary Documentation Release: provides more complete and accurate information about the product, but is not in final format.
- FDR Final Documentation Release: a complete production description: edited, formatted and produced at a high standard of graphic quality.
- MAN Manual: early reference documents to be phased out by PDR's and FDR's.
- PTU Prime Technical Update: interim updates to existing documents.

Copyright 1977 by  
Prime Computer, Incorporated  
145 Pennsylvania Avenue  
Framingham, Massachusetts 01701

The information in this document is subject to change without notice and should not be construed as a commitment by Prime Computer Corporation. Prime Computer Corporation assumes no responsibility for any errors that may appear in this document.

First Printing November 1977

## CONTENTS

<u>Section</u>	<u>Title</u>	<u>Page</u>
	<u>PART 1 TUTORIAL</u>	
SECTION 1	INTRODUCTION	1-1
	PURPOSE	1-1
	ORGANIZATION	1-1
	TERMINALS AND COMPUTERS	1-3
	WHAT YOU DON'T SEE	1-3
	HOW COMPUTERS THINK	1-5
	TALKING WITH YOUR COMPUTER - SOME BASIC TERMS	1-6
	CONVENTIONS USED IN THIS MANUAL	1-7
	YOUR TERMINAL KEYBOARD	1-10
SECTION 2	HI - I'M PRIMOS	2-1
	YOUR USER FILE DIRECTORY	2-1
	HOW TO CORRECT TYPING ERRORS AT THE TERMINAL	2-2
	HITTING THE CARRIAGE-RETURN KEY INPUTS A LINE FROM THE TERMINAL	2-2
	GETTING READY TO LOG IN	2-2
	CONNECTING THE TERMINAL WITH THE COMPUTER	2-3
	LOGGING IN	2-4
	PROBLEMS IN LOGGING IN	
	OTHER REASONS YOU CAN'T LOG IN	2-6
	FINDING OUT WHAT'S IN YOUR UFD	2-7
	LOGGING OUT	2-8
	NOTES ON LOGGING OUT	2-8
SECTION 3	THE ESSENTIALS OF EDITOR	3-1
	WHAT EDITOR IS	3-1
	CONVENTIONS IN EDITOR	3-1
	HOW EDITOR WORKS	3-3
	INPUT AND EDIT MODES	3-3
	ENTERING EDITOR	3-3
	ENTERING TEXT IN INPUT MODE	3-4
	SWITCHING FROM INPUT TO EDIT MODE	3-5
	HOW EDITOR WORKS ON ITS FILE	3-6
	GIVING COMMANDS IN EDIT MODE	3-7
	SWITCHING FROM EDIT TO INPUT MODE	3-8
	BASIC EDITOR COMMANDS	3-8
	EDITOR'S ERROR MESSAGES	3-9
	PRINTING COMMANDS	3-9
	POINTER-MOVING COMMANDS	3-11
	LINE CHANGING COMMANDS	3-14
	ENDING AN EDITOR SESSION	3-18
	MISCELLANEOUS INFORMATION CONCERNING EDITOR	3-22
	EDITOR'S OTHER COMMANDS	3-25

## CONTENTS (Cont'd)

<u>Section</u>	<u>Title</u>	<u>Page</u>
SECTION 4	MORE PRIMOS	4-1
	USING OTHER UFDS: THE ATTACH COMMAND	4-1
	MAKING SUB-UFDS: USING THE CREATE COMMAND	4-2
	LOOKING AT YOUR FILES USING THE SLIST AND SPOOL COMMANDS	4-4
	RENAMING AND DELETING FILES AND SUB-UFDS	4-7
SECTION 5	THE ESSENTIALS OF RUNOFF	5-1
	INTRODUCTION	5-1
	CREATING THE SOURCE FILE	5-2
	RUNOFF COMMANDS	5-2
	RUNOFF COMMAND CONVENTIONS	5-4
	TYPES OF COMMANDS	5-4
	PAGE-FORMATTING COMMANDS	5-5
	LINE-FORMATTING COMMANDS	5-7
	RUNNING RUNOFF (PROCESSING YOUR SOURCE FILE)	5-16
	RUNOFF COMMAND ERRORS	5-20
	DONE: THE PROCESSED OUTPUT FILE	5-22
SECTION 6	MORE RUNOFF	6-1
	INTRODUCTION	6-1
	PAGE-FORMATTING COMMANDS	6-1
	OUTPUT OPTIONS	6-4
	SPECIAL CHARACTER, SYMBOLS AND CONVENTIONS	6-7
	BLOCKS OF TEXT, ART AND INSERTED FILES	6-13
	INDEXING	6-14
SECTION 7	RUNOFF DECIMALIZATION	7-1
	INTRODUCTION	7-1
	DECIMALIZATION COMMANDS	7-2
	TABLE OF CONTENTS	7-6
	DECIMALIZATION COMMAND SUMMARY	7-8
	TABLE OF CONTENTS COMMAND SUMMARY	7-9
<u>PART 2 REFERENCE</u>		
SECTION 8	THE EDITOR REFERENCE SECTION	8-1
	EXPLANATION OF THE COMMAND FORMAT	8-1
	COMMAND FORMATS	8-3
SECTION 9	THE RUNOFF REFERENCE SECTION	9-1
	EXPLANATION OF THE COMMAND FORMAT	9-1
	COMMAND FORMATS	9-2

CONTENTS (Cont'd)

<u>Section</u>	<u>Title</u>	<u>Page</u>
	<u>INDEXES</u>	
	SUBJECT INDEX	X-2
	CROSS-REFERENCE OF PRIMOS COMMANDS	X-13
	CROSS-REFERENCE OF EDITOR COMMANDS	X-14
	CROSS-REFERENCE OF RUNOFF COMMANDS	X-17
	INDEX OF SYSTEM MESSAGES	X-21

## ILLUSTRATIONS

<u>Figure</u>	<u>Title</u>	<u>Page</u>
1-1	What You See, What You Don't See	1-4
1-2	Two Typical Keyboard Layouts	1-12
2-1	Sample PRIMOS Session	2-9
4-1	Advanced PRIMOS Session	4-10
5-1	Sequence of Using RUNOFF	5-3
5-2	RUNOFF Default Page Specifications	5-6
5-3	Sample RUNOFF Report	5-17

PART 1  
TUTORIAL



## SECTION 1

## INTRODUCTION

## PURPOSE

THE NEW USER'S GUIDE TO EDITOR & RUNOFF has been written for the new computer user who may not have had any prior experience in working with computers or text-processing systems. We'll tell you exactly what you need to know to immediately log in to the computer and use EDITOR and RUNOFF.

EDITOR is a text-editing system. Using EDITOR, you can type text into the computer, edit it, and save it for later use.

RUNOFF is a formatting system for printed text. With just a few simple commands, you can turn your typed input into pages which are neatly arranged in whatever manner you wish.

## ORGANIZATION

The contents of this manual are:

- Section 1: INTRODUCTION (this section). Gives general information about computers, terminals, and the various conventions used in this manual.
- Section 2: HI - I'M PRIMOS. Gives you basic information about how to use a Prime computer.
- Section 3: THE ESSENTIALS OF EDITOR. Introduces you to EDITOR, and teaches you enough about EDITOR to do most jobs.
- Section 4: MORE PRIMOS. Additional information about Prime computers, which you will not need until after you have been using EDITOR.
- Section 5: THE ESSENTIALS OF RUNOFF. Teaches you enough about RUNOFF so that you can use it for a number of standard tasks.
- Section 6: MORE RUNOFF. Explanations of how to do slightly more complicated or non-standard work using RUNOFF.
- Section 7: RUNOFF DECIMALIZATION. How to do decimalized headers and tables of contents.
- Section 8: EDITOR REFERENCE SECTION. Full information on all the EDITOR commands, in alphabetical order.

Section 9: RUNOFF REFERENCE SECTION. Full information on all the RUNOFF commands (except for the Decimalization commands) in alphabetical order.

Appendix A: COMMAND SUMMARIES. Summarizes the PRIMOS, EDITOR and RUNOFF commands, and error messages for easy use.

We've arranged the material so you can log in to the computer and begin working almost immediately. And the further you read, the more you'll be able to do.

Once you've become familiar with the computer and have had some practice using EDITOR and RUNOFF, you should have little trouble learning to use the more advanced techniques and more powerful commands. Then, when you are an experienced user, you can keep the command summaries by your terminal, and turn to the reference sections whenever specific questions arise.

## TERMINALS AND COMPUTERS

It is very unlikely that you are near your Prime computer at the moment. The computer is off somewhere in the computer room, sitting in its cabinet, alongside a couple of disk drives, tape transports, line printers, teletypes, and so on, all under the watchful eye of a computer operator person or two. All you need to talk with the computer is your terminal, and -- maybe -- a telephone coupler.

Prime computers can often be connected with up to 63 working terminals at the same time.

Some terminals connect directly to the computer by a direct length of wire. Other terminals communicate with the computer over the telephone, using small devices called acoustical couplers which translate the computer's and terminal's interaction into signals which can be sent through a phone line.

The top half of Figure 1-1 illustrates those aspects of the computer system which you are likely to see and use - a terminal, telephone and acoustic coupler (maybe), and sometimes the line printer.

## WHAT YOU DON'T SEE

As far as you're concerned, the computer is a large black box which always behaves in the manner described in this book. The bottom half of Figure 1-1 shows one way you can picture what the computer does.

Think of the computer as a company which rents desk space plus office services. One section of the office is a storage area containing a virtually unlimited number of file cabinets, plus a well-stocked shelf containing office equipment.

The office is run by an office manager, whose name is PRIMOS. There is a specific list of jobs which PRIMOS will do for you; PRIMOS will not understand anything you say that is not on this list. These jobs are explained in manuals such as this one. (There are two types of jobs: those which have PRIMOS perform a service for you, and those which have PRIMOS get you something from the supply shelf.

The supply shelf contains the office equipment available to all users. Unlike real offices, there is never any problem with getting a particular piece of equipment at the time you ask for it, no matter how many other people are using one at the same time. This manual explains how to use the "pieces of equipment" called EDITOR and RUNOFF; other manuals explain other equipment.

People who wish to work in this office identify themselves by giving a piece of identification called a user-name. Many people may know a particular user-name; some people will know several different user-names. With some of these user-names, you will have to give a password; this prevents unauthorized people from using these names.

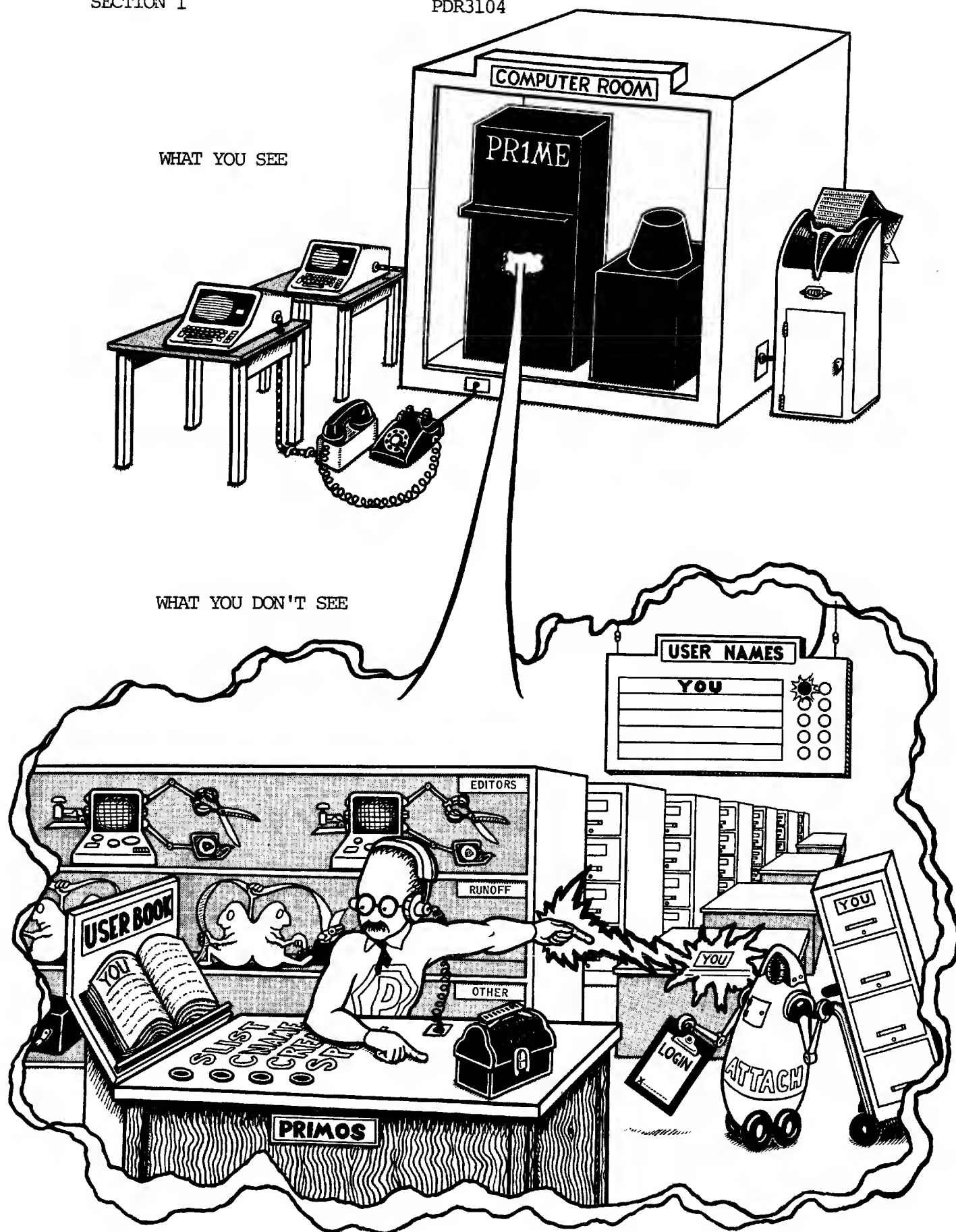


Figure 1-1. What You See, What You Don't See

There is a labelled file drawer for every user-name. In your drawer are folders; each folder contains a file. Your drawer is always large enough to hold all the files you need. PRIMOS allows you to label each folder.

PRIMOS keeps a notebook, containing a section for every user. When somebody asks PRIMOS for a desk to use, PRIMOS flips through this notebook to make sure the user is a good account. If this is the case, PRIMOS assigns you to the first available desk, signs you in, and drags over your file drawer. Once you have been signed in, PRIMOS will get you any other file drawer which you are entitled to use. Any number of people can work out of a file drawer at the same time, although sometimes only one person can work on a specific file at a time.

You can do any number of things at your desk by using the office equipment: make new files, change them, do computations, move files around, look at files, run various programs. All you do is tell PRIMOS what you want to do, and PRIMOS will take care of the precise details. Again, you must specify exactly what it is you want to do.

When you are done working, PRIMOS puts away all your files, closes up the drawers, returns the supplies to the shelf, and tells you the total amount of work you did. At this point, the desk is clean, and ready for its next occupant; you are done.

#### HOW COMPUTERS THINK

Computers are machines. They are complicated machines, full of large quantities of little widgets - but just as you don't have to know how an internal combustion engine works to drive a car, you don't need to understand how the computer works, or why; what you do need to know is the rules of the road - how to use it.

That's what this guide does - it's a road map/instruction handbook to get you where you want to go.

Like any other machine, computers do not really "think", they just follow orders and do EXACTLY WHAT YOU TELL THEM TO DO. This means you have to tell the computer PRECISELY what you want. If you make a mistake that still can be understood, it will follow the order anyway. Computers do exactly what they are told. This means, of course, you have to know exactly what you want to do, in terms which the computer can understand. That is what this manual will teach you. Since the computer follows your orders precisely, you obviously want to avoid making mistakes.

To help you do this, the computer continually checks everything you tell it to make sure you have given it reasonable instructions, -- so far as it can tell. It knows what each possible command is supposed to look like in terms of spelling, permissible abbreviations, and, occasionally, whether the data associated with the command is of the appropriate type.

You even have a chance to correct typing errors and redo entire lines before they are read. If the system is not happy with what you have told it, for some reason, it will tell you by printing out one or more Error Messages. Usually these messages are clear, and you will have no trouble figuring out what has happened. If you are confused, check the list in the back of this manual or ask the System Administrator.

What the computer cannot determine, however, is whether what you said is what you meant to say. Always keep in mind:

- Computers have no common sense. No matter how strange your request may sound, if the system can do it, it will.
- Computers are always consistent. If the circumstances are the same, the computer will always react exactly the same way to the same command (or mistake).
- Nothing breaks when you make mistakes. Computers are built to be used by people. It's not hard to make little mistakes. It is possible, though difficult, to scramble your own files. However, it is very difficult to do something that cannot be undone, and it is impossible to do anything at your terminal that actually harms the computer.

So, with a little practice, you'll be able to analyze what it is you want to do and then determine how to make the computer do it.

#### TALKING WITH YOUR COMPUTER - SOME BASIC TERMS

A given period of time spent at your terminal working with the computer is called a session.

All work in a session is done on files. A file is a series of letters and numbers arranged in lines and which has some meaning to you.

The conversation which takes place between you and the system is called dialogue, or interactive dialogue (because you and the system are interacting).

Your portion of the dialogue is called input. Input consists of commands and data.

A command is a line of input consisting of a command word, which specifies what to do, and parameters, which are what the command is done to (or with). For example, in the command: COOK dinner, COOK is a command word and dinner is a parameter. Commands do things like:

- Request information, such as the names of the files in your User File Directory or a listing of the contents of a file.

- Specify a given system or program you want to use, such as EDITOR or RUNOFF
- Manipulate your files--rename, print out and delete them, etc.
- Do specific tasks--change a line, find a word, indicate you're about to type in some data, etc.

Data is the actual information the system processes, as directed to by the commands. Data consists of numbers and text strings.

There is a standard way that many commands are performed. To save trouble, the computer will automatically assume you want these commands done in these specified ways unless you explicitly tell it otherwise. These choices are called defaults - they are made by default if you don't say differently. All defaults are clearly defined in this document when each command is explained.

Dialogue from the system is called output. The various types of output are:

- Requested information -- Information you asked to see.
- Reminders -- Prompts, queries and verifications, to help you get everything the way you want it.
- Messages -- Error messages and messages from the System Administrator to certain or all users.

### Error Messages

The computer gives you an error message:

- Whenever you try to do something that is "illegal" -- it cannot be done.
- Whenever you type in a command that makes no sense.

Depending on which of these categories a given error will fall into, you will either get a question mark (?), meaning "that makes no sense," or an error message containing a very abbreviated explanation of what the system thinks you did wrong, plus the word "ER!" (for "Error"). These messages refer to a table of explanations. The error will probably be caused by a typing error, or by your losing track of what precisely you had to do. Look at whatever you had typed that provoked the error, and try again.

### CONVENTIONS USED IN THIS MANUAL

Commands may contain command words, parameters, and keywords. Commands are given in a general Command Format, as follows:

COMMAND parameter

Command Words

The Command Word is given in upper case; however, you may input command words in any combination of upper and lower case characters. You do not always have to type the entire command word; the underlined letters are sufficient. (You may type any of the non-required letters, in addition, - e.g., for PRINT, you could say P, PR, PRI, prin, or PRint.)

Parameters

Commands often have parameters, which specify something about how the command should be performed. For example:

PRINT n

means "Print n lines", where n is a number.

Parameters are written in lower-case letters. If a parameter is enclosed in square brackets, like so - [parameter] - it is optional. For example, if a format is PRINT [n]; you could say PR5, PR-10 or PR (note: these do not mean the same thing). If you do not specify a parameter, the default value is used.

There are two types of parameters: Numeric and text.

A numeric parameter is represented in Command Format by a lower-case letter, usually "n". Numeric parameters specify things like numbers of lines, times to do an operation, etc. As an aid, all parameters are underlined when they appear in text, like this: "...n lines from filename." The value you give n can always be positive; often n can be zero or negative (the explanation of the command will specify).

If n is negative, the number must have a minus sign (-) immediately to its left; e.g., -12, -5 with no intervening spaces.

In most cases, parameters have default values, which are assumed, if you omit them from the input.

In the Command Format, when n is enclosed in parentheses ( ) immediately following a command word, e.g., FIND(n) - both parentheses are required, with no space between the command word (or abbreviation) and the left parenthesis. For example:

FIND(6)  
F(2)  
FIND(5)

A text parameter is one or more characters, sometimes including blanks in a row. A text parameter can either be:

- a filename
- a character
- a string



A filename is the name of a file - e.g., MEMO, DAVIDS-NOTES.

A character is a single character; the value of a character may be a letter, number, or symbol.

A string is a series of characters which has no command meaning to the computer, but has meaning to the user like "PAGE 99" or "PLEASE RETURN AT ONCE". A string that contains no characters is called a null string.

### Keywords

Keywords are words in upper-case letters other than the command word. They are important in clarifying the use of the parameters. When they appear in a format, keywords must always be used exactly as they appear. There are not very many keywords used in this manual; the most common one is TO.

### Convention in the Examples

When examples have been taken from actual computer sessions, the user's input has been underlined, and the computer's output has not.

## TERMINALS

A terminal is a machine that allows you to "talk" with the computer. It must have a keyboard (to let you type your input) something on which it can display output, and a connection to the computer.

It is important to remember that you do not have to always use the same terminal to work on the computer. Any terminal that is connected to the computer will do, so long as it is the proper kind for what you want to do.

### Type of Terminals

There are two basic types of terminals - ones that type output onto paper, and ones that display output on a TV screen.

Terminals that type are referred to as "hard-copy" terminals. These are useful when you want to save a printed copy of your session, or type out a particular file.

Terminals with screens are called either video terminals or CRT (for Cathode Ray Tube) terminals. Video terminals produce output faster than hard-copy terminals. They are also useful when you are doing a lot of work for which the end result is important but not the in-between stages. Working on a video terminal allows you to not consume lots of paper which you would only dispose of later.

## YOUR TERMINAL KEYBOARD

Basic Layout

The layout of the terminal keyboard can vary from one type of terminal to another. Figure 1-2 shows two typical keyboards.

Besides the usual letter, number and punctuation symbols, your terminal also has a variety of special symbols and keys. The number and letter keys are arranged in the same positions as on all standard typewriters.

The punctuation marks may be located on different keys, depending on the terminal model.

Special keys fall into the following categories:

- Terminal Controls & Switches
- Special Keys
- Special Characters

Terminal Controls & Switches

Terminal controls and switches affect the special ways a terminal performs. Depending on what model terminal you have, these may be switches on the front, side or bottom of your terminal; they may be keys on the side of your keyboard; they may be differently-colored keys to the side of the keyboard.

The controls and switches that you need to know about are:

- |            |  |
|------------|--|
| ON/OFF     | This is the power switch. Printing terminals often have an indicator light which glows when the power is on. On some terminal models, this switch is located on the bottom of the terminal.                        |
| LINE/LOCAL | This switch controls whether or not the terminal is sending input to the computer. In LINE Mode, the terminal and the computer are connected; in LOCAL Mode, the terminal acts just like a specialized typewriter. |

This switch is often labeled:

ON-LINE/OFF-LINE  
REMOTE/LOCAL  
LINE (and an indicator light which is ON in LINE Mode,  
OFF in LOCAL Mode)

A simple way to determine which mode your terminal is in is to press the Carriage-Return Key. If the terminal advances to a new line, it is in LINE Mode; if it simply returns to the beginning of the same line, it is in LOCAL Mode.

UPPER-CASE/  
LOWER-CASE      Unlike the SHIFT Key, the UPPER-CASE/LOWER-CASE Key only affects the meanings of the letter keys. UPPER-CASE causes all letters to print in upper-case, no matter what the setting of the Shift Key is; LOWER-CASE lets you select between upper- and lower-case in the standard manner (by using the Shift Key).

On some terminals, this switch is located on the bottom, instead of on the keyboard.

This key is often labeled:

CASE  
UPPER/LOWER  
U/C    (for Upper-Case - when on)

Certain terminals do not have this switch at all; they produce only upper-case letters.

### Special Keys

The CONTROL Key: The key labeled CONTROL (or CNTRL) is used to indicate Control Characters by pressing it at the same time as a character key. These have particular meanings to the computer which are quite different from the character's normal meaning.

The RUBOUT Key: The key labeled RUBOUT has a special use in RUNOFF. This key has no effect anywhere else in your Prime computer.

### Special Characters

Up-Arrow: The character indicated as an Up-Arrow (↑) or a carat (^) is called Up-Arrow. The Up-Arrow has certain specific uses in the EDITOR and RUNOFF.

The Backslash: The Backslash symbol (\) looks like a backwards slash. The backslash has a specific meaning in EDITOR - the default TAB character.

If you have difficulty getting your terminal to work, ask for help.

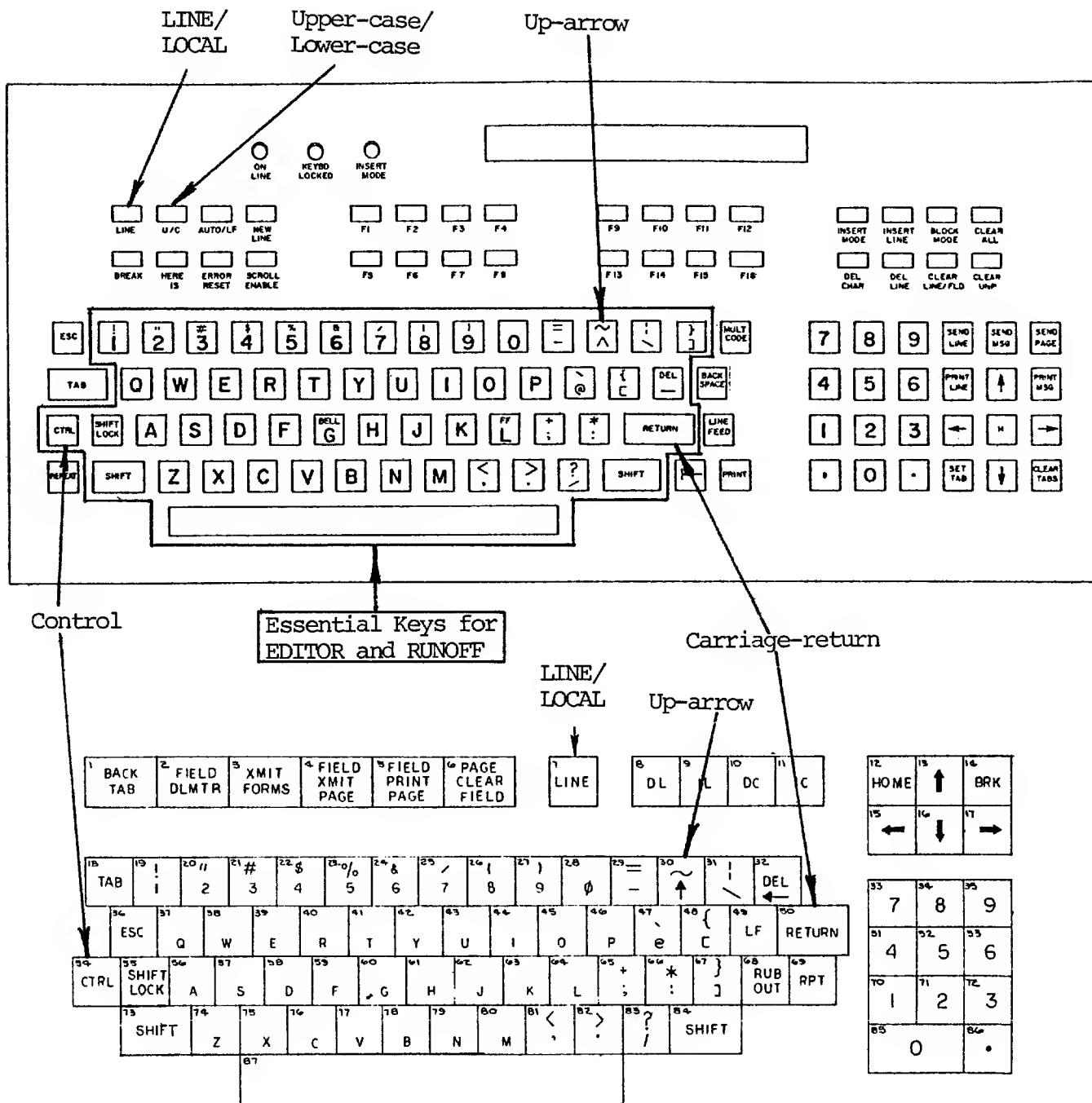


Figure 1-2. Two Typical Keyboard Layouts

## SECTION 2

## HI - I'M PRIMOS

PRIMOS is the operating system of your Prime computer. A computer's operating system (also called the Monitor, Monitor System, or Supervisor) acts as Security Guard, Time Administrator, Bookkeeper, File Clerk, Messenger, Runner, Lifeguard, Information Operator and Office Manager. PRIMOS takes care of all these necessary chores so that you can use the computer to do what you want without worrying about how it's being done.

PRIMOS' chores include:

- Keeping track of who's allowed to use the system.
- Keeping track of who's doing what on the computer at any given time, from what terminal they're doing it, and how much computer time and space they're using.
- Giving you the appropriate "piece of equipment" - e.g. EDITOR or RUNOFF.
- Doing all the house work to clean up behind you.

#### YOUR USER FILE DIRECTORY NAME

In order to do any work on the computer, you must first tell PRIMOS to connect you with your User File Directory.

PRIMOS recognizes each user by the name of the account they type in. These accounts are called User File Directories (UFDs); their names are UFD-names. In order to do any work on a Prime computer, you must know a UFD-name. For each account name, there is a corresponding file area. All work you do will be charged to your account. You may, however, also use files which are not in your own UFD, if you know the name of the UFD they are in.

#### Passwords

In order to keep just anyone from getting access to each person's UFD, PRIMOS can be told to require a password from a prospective user. Knowing the password identifies you as someone authorized to use the associated UFD-name.

#### How to Get Your UFD

Go to your System Administrator and ask for the name of a UFD, and a password. (Your UFD-name is often the name of your department or project, or your own name. Depending on circumstances, you may share a UFD with other people -- or have several UFDs of your own.)

## HOW TO CORRECT TYPING ERRORS AT THE TERMINAL

For every time you type the Erase character, you will erase the most recent un-erased character. Unless your System Administrator has changed it, PRIMOS' Erase character is the double-quote (").

Each " you type rubs out the most recently typed character other than another double-quote mark. So typing two double-quotes ("" ) would rub out the last two characters, three double-quotes (""") would rub out three, and so on. In other words, if you had typed lygin""""ogim"n, the computer would read simply login.

The Kill Character will erase the entire line. Unless your System Administrator has changed it, PRIMOS' Kill character is the question mark (?). So for example,

login my filp?login myfile

would be read

login myfile

## HITTING THE CARRIAGE-RETURN KEY INPUTS A LINE FROM THE TERMINAL

The computer pays no attention to what you type on the terminal until you hit the Carriage-Return Key. When you press the CR key; the system reads the line of input, if any, from your terminal and:

- Inspects the line for typing corrections (Erase & Kill).
- Analyzes what you have typed. If your input can be interpreted as a command, PRIMOS does it; if not, it gives you an error message, which is an analysis of what it interpreted your command to be, and why it wasn't acceptable.

Remember: You input a line of typing from the terminal to the computer by hitting the Carriage-Return Key.

## GETTING READY TO LOG IN

Logging In identifies you to the system. It gives you access to a work area, your own files, plus all the general resources of the computer - including EDITOR and RUNOFF, of course.

In order to log in, you must know the name of a User File Directory (UFD). This is an identifying phrase which the System Administrator has assigned to an account in the computer. You may also need to know a password to go with this UFD; this will identify you as an authorized person. Not all UFDs will require passwords.

The sequence of steps involved in logging in is:

1. Connect terminal to computer.
2. Give LOGIN command.

#### CONNECTING THE TERMINAL WITH THE COMPUTER

If your terminal is not connected to the computer, you'll never be able to log in. So this is clearly the first thing you must get done.

#### Turning Terminal On

Plug the terminal in and turn the power on. The power switch on your terminal may be located either on the keyboard or on the bottom of the terminal. Keyboard switches or keys are usually labelled POWER or ON/OFF. When you turn your terminal on, a small indicator light often goes on.

#### Line Mode

As we explained in the Introduction (Section 1), there is a key or switch somewhere on your terminal called the Line/Local Switch. This allows you to use your terminal either as an independent machine or with the computer. In order to connect the terminal to the computer, you must be in Line Mode. This switch or key is somewhere on 'the keyboard', or on the front or back panel. It may also be labelled:

LINE  
REMOTE/LINE  
ON-LINE/OFF-LINE

The terms LINE, REMOTE, and ON-LINE all mean the same thing.

Make sure this switch is properly set. (Some terminals also have an indicator light which goes on in Line Mode.)

#### Connect Terminal To Computer

Some terminals are connected directly to the computer, by wires which run between them.

Other terminals become connected to the computer over the telephone. If the latter is the case, in order to connect your terminal up, you will need the following:

1. An Acoustic coupler This converts the terminal and computer signals into sounds which can pass through the telephone lines. Some terminals have acoustic couplers built into them, on the back or side; these built in couplers look like a pair of holes into which you can insert a phone handset. Other terminals will have a small acoustic coupler nearby.

2. A telephone. Any regular telephone will do; however, since the odds are good that you will want to use the phone for this purpose for some time, you want to make sure that nobody else can pick up an extension of this line, and that there other phone lines available for you and other people to use. People who use their phone lines for their terminals frequently tend to have a special phone line put in for this purpose.
3. A phone number for the computer. Obviously, you can't call the computer unless it, too, has a telephone connection. You should have a least one phone number that connects directly to the computer. There may be several numbers, depending on how far away the computer is.

If you have these three things, you are ready to dial in.

### Dialing In

Dial the phone number for the computer system on the telephone. Listen in the earpiece after you finish dialing. If you get a busy signal, this means all the computer's phone lines are busy. Hang up and try again later.

If the lines are not busy, the ring will stop after one or two rings, and you will hear a high-pitched tone in the earpiece of the phone handset. This means the computer is on the line. Place the telephone handset correctly onto the openings in the acoustic coupler.

If your terminal has a LINE indicator light, it should light up. Press the CR Key once or twice.

Now that your terminal is connected to the computer, you're ready to actually log in.

### LOGGING IN

If you have been given a UFD-name, and have connected the terminal to the computer, you are ready to give the LOGIN command.

The format of the LOGIN command is:

```
LOGIN ufd-name [password]
```

If you have done this correctly, PRIMOS will acknowledge you as being logged in, like so:

```
login sales
SALES (15) LOGGED IN AT 10'36 110579
```



The number in parenthesis is your Job Number assigned by PRIMOS. The time you logged in (here 10'36) is expressed in the 24-hour system, also known as Military Time. Here are a few examples of 24-hour times and their 12-hour equivalents.

0'00	Midnight
0'01	12:01 A.M.
1'00	1:00 A.M.
3'39	3:30 A.M.
11'59	11:59 A.M.
12'00	Noon
12'01	12:01 P.M.
13'00	1:00 P.M.
15'30	3:30 P.M.
23'59	11:59 P.M.

The current date is expressed as Month Day Year. For example, the number "110579" means November 5, 1979.

#### PROBLEMS IN LOGGING IN

It is quite possible that for some reason you will not be able to log in every time you try. This is most likely due to some small typing error on your part, and is easily remedied. As a rule, hitting the Carriage-Return Key and simply typing the LOGIN command again will do the trick.

Here's a list of the possible error messages you may get when trying to log in, along with their meanings and what you should do:

<u>MESSAGE</u>	<u>MEANING(S)</u>
LOGIN PLEASE	<ul style="list-style-type: none"> <li>● You misspelled the word LOGIN. <u>Try again.</u></li> <li>● You tried to start work without logging in. <u>Try again.</u></li> <li>● There was a temporary transmission problem in the line. <u>Hit two or three keys (any keys) and a CR, then try again.</u></li> </ul>
NOT FOUND	<ul style="list-style-type: none"> <li>● You misspelled the name of your UFD. <u>Try again.</u></li> <li>● No UFD by that name exists. <u>Check your UFD name.</u></li> </ul>
ILL REM REF.	<ul style="list-style-type: none"> <li>● Same reasons as NOT FOUND (above).</li> </ul>
NO RIGHT(LOGIN)	<ul style="list-style-type: none"> <li>● You forgot to type your password. <u>Try again.</u></li> </ul>
RESERVED CHARACTER	<ul style="list-style-type: none"> <li>● You accidentally typed a character which may <u>not</u> be used in a command. <u>Try again.</u></li> </ul>

- |                            |   |
|----------------------------|---|
| BAD PASSWORD<br>(ATCH\$\$) | ● You misspelled your password - or the password has been changed. <u>See your System Administrator.</u>  |
| NOT A UFD                  | ● You tried to log into something which can't be logged into. <u>Try another ufd-name. If it was your ufd-name, and you are sure you spelled it right - try a few more times to be sure - call your System Administrator.</u> |

#### OTHER REASONS YOU CAN'T LOG IN

There are also a number of physical circumstances that can prevent you from logging in, none of which will yield Error Messages. These are:

1. Your terminal is not turned on. (Remember: Some terminals have the power switch located on the bottom; carefully lift terminal up and look.) Turn terminal on. Make sure the power cord is properly connected and that the terminal is plugged in.
2. The terminal is in LOCAL mode. This means it is acting purely as a typewriter. If you have a LINE/LOCAL switch, push it. Then hit a key or two - any key - plus the CR key once or twice; if you've succeeded, the system will respond with the message LOGIN PLEASE.
3. The settings inside your terminal are inappropriate. See your System Administrator.
4. Your terminal is physically disconnected from the computer, in any of the following ways:
  - Somebody pulled or cut the wire. Call the repair person.
  - If your terminal connects by way of a telephone, check the following:
    - a. Is your Acoustical Coupler ON? If not, turn it on.
    - b. Did you dial the correct number for the computer? (If you did, you'll hear a high-pitched whistle in the earpiece.) Try calling again.
    - c. Is the phone handset properly inserted? Make sure the earpiece is over the part of the coupler that makes noise, the mouthpiece and cord by the other part. Make sure the handset is snugly inserted.
5. All the phone lines to the computer are busy. Try again. If no luck, wait a few minutes, then try again.
6. Your terminal is not working. Call the repair person.

7. Last, but not least, the whole computer may be "down". In other words, OFF. If this is true, there's nothing you can do but wait. The system may not be working for a short time. If your terminal is directly connected, hit the CR key every so often to see if the system has come "up" (been turned on) again.

When you have successfully logged in, you'll get the message:

```
ufd-name (job or term #) LOGGED IN AT time date
```

OK,

If for any reason, you cannot succeed in logging in after a reasonable number of attempts, consult your System Administrator.

#### FINDING OUT WHAT'S IN YOUR UFD

Now that you're logged in, you've got a "work space" plus a User File Directory containing files to work on. (Unless you have been given a brand-new UFD, of course. In this case, there are no files in your UFD -- it is "empty".)

In order to specify which file to work on, you have to know what files are in your UFD. While you could keep track of the contents of your UFD by writing down the names of files on a sheet of paper and keeping it with you all the time, PRIMOS can do this record-keeping for you faster and more reliably. PRIMOS maintains an up-to-date list of the names of all the files within your directory. To see this list, you give the LISTF (for List Files) command. The format of the LISTF command is:

LISTF

You need only type the letter L, if you wish.

Let's suppose you have logged into a UFD named STAFF, which contains the files FRANK, MARTHA, ALBERT and ROSEMARY. Giving the LISTF command would have this effect:

```
listf
```

```
UFD=STAFF 3 0
```

```
FRANK  MARTHA  ROSEMARY      ALBERT
```

OK,

If there were no files in the STAFF UFD, doing a LISTF would have this result:

```
OK, listf
```

```
UFD=STAFF 3 0
```

```
.NULL.
```

OK,

The number after the name of the UFD helps the computer tell where the UFD is. This number is not important to you. The letter by the number will either be O (for Owner) or N (for Non-owner). Owner means you gave the right password, so you can inspect and change files; Non-owner means you did not give the password necessary to be able to change anything in the UFD, but you are permitted to inspect files.

It's a good idea to give the LISTF command at the beginning and end of each session, to check up on what files you've got in your directory.

## LOGGING OUT

When you have finished a session at the terminal, you must inform PRIMOS you are done. This is called logging out.

Logging out is the opposite of Logging in -- it tells PRIMOS that you have finished whatever you were doing (or are going to stop for now anyway). PRIMOS puts all your files away, and signs you out.

You log out by giving the LOGOUT command. The format of the LOGOUT command is:

### LOGOUT

PRIMOS acknowledges the command with the following message:

```
UFDname (user #) LOGGED OUT AT (time) (date)
TIME USED = Terminal time  CPU time  I/O time
```

- Terminal time is the amount of elapsed clock time between logging in and logging out. It tells you how long your session lasted. Most of your 'real' time the computer probably was "waiting" -- i.e., doing other work until you gave it a new command.
- CPU time means how much actual time the computer spent following your commands. (CPU stands for Central Processing Unit.)
- I/O time is the amount of time the computer spend moving data around.

Giving the LOGOUT command looks like this:

```
OK, logout
STAFF (15) LOGGED OUT AT 11'28 110579
TIME USED= 0'12  0'06  0'032
```

## NOTES ON LOGGING OUT

It's often a good idea to log out if you're going to leave your terminal for awhile, to prevent anyone else from doing work on your account time, or from making changes in your files. There's nothing wrong with leaving yourself logged in all day, but since logging in and out is so easy, it's a good idea to only be logged in when you want to work

```
login staff
STAFF (15) LOGGED IN AT 10'36 110579

OK,listf

UFD=STAFF 3 0

FRANK  MARTHA  ROSEMARY      ALBERT

OK, logout
STAFF (15) LOGGED OUT AT 11'28 110579
TIME USED= 0'12  0'06  0'032
```

Figure 2-1. Sample PRIMOS Session

## SECTION 3

## THE ESSENTIALS OF EDITOR

## WHAT EDITOR IS

EDITOR is a system designed to let you create and edit text files on the computer. You make a file by typing it on your terminal, line by line, then editing this text.

If you have a file typed in like this

```
'Twas broiling time, and the lithe and slimey toves
Did gyre and gimbol
It was many and many a year ago
And the mome raths outgrabe.
All mimsey were the borogroves
```

EDITOR lets you correct and change it to, for example:

```
'Twas brillig, and the slithey toves
Did gyre and gimbol in the wabe.
All mimsey were the borogoves
And the mome raths outgrabe.
```

## CONVENTIONS IN EDITOR

Erase and Kill Characters

EDITOR's Erase and Kill Characters are the same as PRIMOS'. Unless you or your System Administrator have changed them, EDITOR's Erase Character is the double-quote (") and Kill Character is question-mark (?).

Command Format

EDITOR's Command Format is:

COMMAND parameter

The word in capital letters is the command word.

Command Words: The underlined letter(s) of the command words indicates the minimum required abbreviations. You must type at least three letters; you may type as many more as you wish. The following are all acceptable ways to input a command whose format is APPEND:

```
A
APP
APPE
APPEND
```

Also, you may type the letters of command words in any combination of upper and lower case letters. All of the following are equally valid:

```
A
a
APpend
append
ApPeND
appEND
```

Parameters: As a rule, the parameter in a command format will be:

- The word filename, representing a filename.
- The letter n, representing a number.
- The word character, representing a single character.
- Any of the words string, text, or newline, representing a piece of text.

Parameters which are enclosed in brackets, like so--[filename]--are optional. EDITOR will use the indicated default value, if you do not specify a value.

If the parameter is a number, represented by the letter n, you do not need to type a space between the command word and the number. For example, the following are all valid:

```
Print5
Pr5
PRINT 15
p-5
p      -5
```

Note that there cannot be a space between the minus sign and a number.

If the parameter is a filename or a character, you must have at least one space between the command word and the parameter, like so:

```
load memo
kill    &
```

If the parameter is a text string ( indicated by text, string, newline) EDITOR assumes that there is exactly one space between the command word (or abbreviation) and the text string and subsequent spaces we considered part of the text string. Like so:

```
find DEPARTMENT
append and so forth
insert Dear Sir or Madame,
```

Any space after the first space is considered part of string. So, for example, the commands:

```
find    henceforth
```

would be using the string " henceforth" which began with three blank spaces.

#### Example Format

Your input is underlined; EDITOR's output is not.

#### HOW EDITOR WORKS

EDITOR has a special file area called the work file which is reserved for its own use. EDITOR puts all input into this work file and does all its editing on the contents of this work file.

#### INPUT AND EDIT MODES

EDITOR has two modes: Input Mode and Edit Mode. In Input Mode, EDITOR treats whatever you type as text which is put directly into your work space, line by line. In Edit Mode, EDITOR treats your input as commands, and executes them using the contents of the work file.

You can switch from one mode to the other without trouble. How you enter EDITOR determines whether you begin in Input or Edit Mode, but in any given session, you are likely to switch between Input Mode and Edit Mode a number of times.

#### ENTERING EDITOR

##### The ED Command

You enter EDITOR by typing the ED command:

```
ED [filename]
```

##### Inputting a New File

If you do not specify a filename when you give the ED command, EDITOR assumes that you want to create a brand-new file. It does not copy anything into its work file; instead, it leaves this file blank, positions the pointer at the top, and goes into Input Mode.



OK, ed  
GO  
INPUT

Everything you type will then be treated as input, and inserted directly into this work space until you give EDITOR the appropriate signal to switch over to EDIT Mode.

### Editing An Existing File

If you do specify a filename when you give the ED command, EDITOR assumes you want to be in EDIT so you can edit this file, e.g.,

OK, ed memol  
GO  
EDIT

Here's what happens when you give the ED command and specify a filename.

1. EDITOR finds this file in your UFD.
2. EDITOR then makes a copy of this file and puts this copy in its own work file. This is called the work file, or EDITOR file. The original file is not affected by this in any way whatsoever.

This process permits you to make changes, alterations, and additions to a copy without having to lose the original file. This way, you don't have to worry if you make a mistake, or change your mind - you can always go back to the original. You can also make several slightly different versions of the same original file.

3. As soon as EDITOR has copied the file into its work space (which takes no noticeable time), it goes into EDIT mode, on the assumption that you want to edit the copied file. If you want to input new text somewhere into the work file, you can switch to Input Mode. Note that inputting new text into the work file does not change the original file.

Before you learn how to switch from Edit to Input Mode, let's take a quick look at how you use Input Mode to enter text.

### ENTERING TEXT IN INPUT MODE

When you are in Input Mode, EDITOR inserts whatever you type into filename (either new or old), line by line. In Input Mode, EDITOR will interpret a typed semicolon (;) as a carriage-return. This allows you to end a line of input by typing either a semicolon or a carriage return, for example, the input:

line one;line two;line three

would become

line one  
line two  
line three

Following a semicolon by another semicolon (in Input Mode) like so:

INPUT  
line one;;line three

inserts a blank line in the file, like so:

line one  
  
line three

This use of the semicolon is particularly handy for inputting a long list of short lines.

INPUT  
apples;bananas;dates;figs;garlic;wolfbane

would become

apples  
bananas  
dates  
figs  
garlic  
wolfbane

This obviously means you cannot type a semicolon as part of the file. Later we'll show you how to get around this.

#### SWITCHING FROM INPUT TO EDIT MODE

EDITOR will switch from Input Mode to Edit Mode whenever you type a null input line. A null input line is a line of input that contains no text or a carriage return. An input line containing only a carriage return is indicated by a semicolon, like so:

;

This is a blank line. A null input line is inputted by typing a carriage return after a semicolon or a carriage return, like so:

	INPUT
	<u>text;(CR)</u>
	EDIT
or	
	INPUT
	<u>text (CR)</u>
	(CR)
	EDIT

Note that the sequence

text ;; text

does not contain a null input line--the second semicolon defines a line of input containing only a carriage return.

You are now in Edit Mode. EDITOR awaits your commands as to how you want to edit the lines of text in the file in its work area.

#### HOW EDITOR WORKS ON ITS FILE

##### Line-Orientation

EDITOR is a line-oriented system. EDITOR sees your file as a collection of lines.

##### The Pointer And The Current Line

EDITOR works on one line of your file at a time. This line is called the current line. EDITOR remembers which line is the current one by positioning a pointer alongside. (Think of the pointer as a paper clip which marks your place but can be moved without leaving marks.)

##### Null Lines

Your file also contains placemarkers called null lines. A null line does not really exist; it marks the place where a line can be inserted.

EDITOR indicates the presence of a null line by outputting:

.NULL.

This is not the contents of the line! It is a reminder.

There is always a null line above the first and below the last line of your file to permit insertion of lines above the top line or below the bottom line. EDITOR has TOP and BOTTOM markers set beyond these null lines, to remind you when you are at the end of you file. Like so:

```
TOP
.NULL.
contents of file
.NULL.
BOTTOM
```

Whenever you delete a line, EDITOR puts a null line opposite the pointer in case you want to insert a line here immediately. The .NULL. will disappear as soon as you move the pointer to a new current line. (Note: A null line is different from a blank line. A blank line has no contents, but has a CR at its end.)

### Line Numbers

EDITOR assigns a list of line numbers to the lines in the work file, for its own reference. This list is constantly updated so that every number belongs to a non-null line.

EDITOR can tell you the line number of the current line, which the pointer indicates. You can move the pointer up and down, either by a given number of lines or by specifying a line number. You can delete or retype the current line, move the pointer to a line which contains a specific word, change, etc.

These tasks are all done by giving commands in Edit Mode.

### GIVING COMMANDS IN EDIT MODE

When you are in EDIT Mode, you type commands instead of text. EDITOR's commands fall into categories according to their type of result. Some of these categories are:

- Commands that print lines (PRINT) or a line number (WHERE).
- Commands that reposition the pointer to a specified line (TOP, BOTTOM, NEXT, POINT) or to a line containing a specified string (FIND, NFIND, LOCATE).
- Commands that change a line (APPEND, CHANGE, DELETE, INSERT, RETYPE)
- Commands that return you to PRIMOS (FILE, QUIT).

The above-named commands are sufficient for you to do basic work with the EDITOR. An explanation of these essential commands follows, shortly.

## SWITCHING FROM EDIT TO INPUT MODE

It is possible to insert a new line into your work file while in EDIT Mode. But if you have several lines to insert in the same place, you will probably find it easier to switch into INPUT Mode.

EDITOR will switch from EDIT Mode to INPUT Mode whenever you type a null command line.

A null command is a command containing no command word or parameter. You input one by following a non-null command with

$$\left\{ \begin{array}{l} \text{semicolon (;)} \\ \text{or} \\ \text{Carriage-Return (CR)} \\ \text{or} \\ \text{comma (c)} \end{array} \right\} \text{ followed by a } \left\{ \begin{array}{l} \text{semicolon (;)} \\ \text{or} \\ \text{Carriage-Return (CR)} \\ \text{or} \\ \text{comma (,)} \end{array} \right\}$$

Except:

Commas do not end the APPEND, INSERT, and RETYPE commands - they are treated as text!!

In other words, typing any pair made of semicolons, carriage-returns, and/or commas after a command, will switch you from EDIT Mode to INPUT Mode, except when the comma follows APPEND, INSERT or RETYPE.

When you switch from EDIT to INPUT Mode, new input will be inserted below the most recent current line in Edit Mode. When you switch from INPUT to EDIT, the last-inputted line becomes the current line.

EDITOR displays the mode whenever you switch between INPUT Mode and EDIT Mode. It is important to remember which mode you are in: When you are in INPUT Mode, trying to give commands will instead put those command words and parameters into your file; in EDIT Mode, EDITOR will try to interpret as commands what you meant as input text. The results in both cases will be undesirable, and not too understandable until you are familiar with EDITOR.

## BASIC EDITOR COMMANDS

You should be able to do most of your text editing using this portion of the EDITOR commands.

APPEND	FIND(n)	POINT
BOTTOM	INSERT	PRINT
CHANGE	LOCATE	QUIT
DELETE	NEXT	RETYPE
FILE	NFIND	TOP
FIND	NFIND(n)	WHERE

These commands are explained in the next few pages. There is a brief summary of the remaining commands at the end of this section. Once you have become proficient at using the first bunch, you should have little trouble learning to use any of these other commands reading the appropriate page in the EDITOR Reference Section (Section 8).

### Sample File

The following file, which is a selection from Lewis Carroll's "Jabberwocky", will be used in all the examples in this section.

'Twas brillig, and the slithy toves  
Did gyre and gimbol in the wabe.  
All mimsey were the borogoves  
And the mome raths outgrabe.

"Beware the Jabberwock, my son!  
The jaws that bite, the claws that snatch--  
Beware the Jub-Jub bird, my son,  
And shun the frumious Bandersnatch!"

He took his vorpal sword in hand,  
Long time the manxome foe he sought;  
And rested he by the tum-tum tree  
And stood awhile in thought.

### EDITOR'S ERROR MESSAGES

In EDIT Mode, if you give EDITOR a command which it cannot understand, you will get one of the following two error messages:

- BAD abbreviation -- This means that you did not use the proper format for the command, assuming you meant to give the command whose abbreviation is given.
- ? -- Your input could not be interpreted as any of the EDITOR commands. This is often a result of thinking that you are in INPUT Mode when you are still in EDIT Mode.

### PRINTING COMMANDS

EDITOR's printing commands are PRINT and WHERE.

#### The PRINT Command

The PRINT command prints *n* lines of your file, including the current line, and makes the last line PRINTed the new current line.

The format of the PRINT command is:

PRINT [n]

If n is -1, 0, or omitted, the default value of 1 is used.

If n is negative, EDITOR moves the pointer back n lines from the current line, and then prints 1 line, which is the new current line.

```

print 3
.NULL.
'Twas brillig, and the slithy toves
Did gyre and gimbol in the wabe.
print 6
Did gyre and gimbol in the wabe.
All mimsey were the borogoves
And the mome raths outgrabe.

"Beware the Jabberwock, my son!
The jaws that bite, the claws that snatch--
print
The jaws that bite, the claws that snatch--
print -5
All mimsey were the borogoves
print -1
All mimsey were the borogoves
print -2
Did gyre and gimbol in the wabe.
print 0
Did gyre and gimbol in the wabe.
print
Did gyre and gimbol in the wabe.

```

The space between PRINT and n is optional. A PRINT immediately after TOP or BOTTOM yields .NULL.

#### The WHERE Command

The WHERE command prints out the line number for the current line.

The format of the WHERE command is:

WHERE

The WHERE command is most useful with the POINT command.

## POINTER-MOVING COMMANDS

Pointer-moving commands either reposition the pointer to a specific line or to a line containing a specific string.

EDITOR's specific pointer-moving commands are TOP, BOTTOM, NEXT, and POINT.

### The TOP Command

The TOP command positions the pointer at the null line at the top of the file, just above the first line of text.

The format of the TOP command is:

```

    TOP

    top
    print
    .NULL.
    print2
    .NULL.
    'Twas brillig, and the slithey toves
    top
  
```

### The BOTTOM Command

The BOTTOM command positions the pointer at the bottom of the file, just below the last line of text.

The format of the BOTTOM command is:

```

    BOTTOM

    bottom
    print
    .NULL.
    next
    BOTTOM
  
```

### The NEXT Command

The NEXT command moves the pointer n lines and prints out the new current line. Positive values of n move the pointer down towards the bottom of the file; negative values up towards the top.

The format for the NEXT command is:

```

    NEXT [n]
  
```



If n is zero or unspecified, the default value of 1 is used. If NEXTing n lines would move the pointer beyond the top or bottom null line, the pointer stops at the null line, and either TOP or BOTTOM is printed.

```

Twas brillig, and the slithey toves
next 2
All mimsey were the borogoves
next 4
The jaws that bite, the claws that snatch--
next -5
Did gyre and gimbol in the wabe.
next 0
All mimsey were the borogoves
next
And the mome raths outgrabe.

```

#### The POINT Command

The POINT command positions the pointer at line n. The line numbers are not actually part of your file; EDITOR generates them for its own reference.

The format of the POINT command is:

```

POINT n

```

The POINT command is equivalent to the sequence TOP, NEXT n. The value of n must be greater than 0. POINT 0 will give you an error message. POINT 1 is equivalent to TOP, NEXT. If n is greater than the number of lines in the file, the pointer will be left at the bottom.

```

point 3
All mimsey were the borogoves
point 7
The jaws that bite, the claws that snatch--
point 2
Did gyre and gimbol in the wabe.
where
LINE      2
point 6
"Beware the Jabberwock, my son!
where
LINE      6

```

#### Note

The command PO 0 yields an Error Message.

### String-Finding Commands

The FIND, NFIND and LOCATE commands reposition the pointer to the first line below the current line which contains the specified string.

These commands distinguish between upper and lower case letters in string. If you are unable to get old lines in your file, but can get newly inserted ones, and your current display is all CAPS, the CASE control on your terminal may be in the wrong position. The string may not contain commas.

#### The FIND Command

The FIND command repositions the pointer to the first line below the current line which begins with string.

The format of the FIND command is:

FIND string

If no line beginning with string can be found, the pointer stops at the end of the file, and the word BOTTOM is printed.

```
find All
All mimsey were the borogoves
print
All mimsey were the borogoves
```

You can also FIND a string starting on other than column 1 of the line, by specifying the number of the column within parentheses directly after the command word, like so:

FIND(n) string

#### Note

The parentheses ( ) around the column number are required. There must be no spaces between FIND and (n).

```
find(5) .mimsey
All mimsey were the borogoves
```

#### The NFIND Command

The NFIND command moves the pointer to the first line below the current line which does NOT begin with string.

The format of the NFIND command is:

NFIND string

```

print 4
'Twas brillig, and the slithy toves
Did gyre and gimbol in the wabe.
All mimsey were the borogoves
And the mome raths outgrabe.
top,next
'Twas brillig, and the slithy toves
nfind Did
All mimsey were the borogoves

```

If NFIND can't find a line which doesn't begin with string, the pointer will be left at BOTTOM.

Like FIND, you can NFIND beginning on a column other than column 1, like so:

```

NFIND(n) string

top,print 5
.NULL.
'Twas brillig, and the slithy toves
Did gyre and gimbol in the wabe.
All mimsey were the borogoves
And the mome raths outgrabe.
next -3
'Twas brillig, and the slithy toves
nfind(5) gyre
All mimsey were the borogoves

```

### The LOCATE Command

The LOCATE command moves the pointer to the first line below the current line which contains string anywhere in that line.

The format of the LOCATE command is:

```
LOCATE string
```

If no line containing string is found below the current line, BOTTOM will be printed and the pointer left at the end of the file.

Example:

```

locate mimsey
All mimsey were the borogoves

```

### LINE CHANGING COMMANDS

The APPEND, CHANGE, DELETE, INSERT, and REYTYPE commands alter the text on one or several lines.

The APPEND Command

The APPEND command attaches the specified string to the end of the current line.

The format of the APPEND command is

```
APPEND string
```

Remember: One blank separates the command word APPEND (or abbreviation) from the string you wish to append. All further blanks will be treated as part of string.

```
print
'Twas brillig and
append the slithy toves
'Twas brillig andthe slithy toves
```

If you want to have one space between the last word of the current line and the first word you are appending, you must type two spaces between the command word and the first word of appended string.

```
print
'Twas brillig and
append  the slithy toves
'Twas brillig and the slithy toves
```

If there is no blank between APPEND and string, EDITOR gives an error message, and you should try again.

Text is terminated by either a Carriage-Return (CR) or a semicolon (;).

You can use commas in the APPEND string, but not semicolons.

```
print
'Twas brillig and
append , dig it, those slithy toves
'Twas brillig and, dig it, those slithy toves
```

The CHANGE Command

The CHANGE command replaces string-1 with string-2 in the following manner: The first character after the command word CHANGE (or abbreviation) will be used as the delimiter in this instance of the command.

The format of the CHANGE command is:

```
CHANGE/string-1/string-2/[G] [n]
```

```

print
And the gnome raths outgrabe.
change/gnome/mome
And the mome raths outgrabe.

```

Any character, including the semicolon (;) and the space ( ) may be used as a delimiter instead of the slash. This means that the following sample commands would all have the same effect.

```

change&gnome&mome&
change ;gnome;mome;
change KgnomeKmome
ch  gnome mome
change >gn>m>

```

If the letter G (for General) is specified, CHANGE will change every occurrence of string-1 on a line. If you don't specify G, only the first incidence of string-1 will be changed.

If the value of n is either 0 or 1, EDITOR will only make changes on the current line. (If n is either 0 or unspecified, the default of 1 is used.)

If a value other than 0 or 1 is specified, EDITOR will inspect and make changes on n lines starting at the current line, and leave the pointer positioned at the nth line. If there are less than n lines in the file below the current line, the pointer will be left at the null line below the last line, and the message BOTTOM will be printed.

EDITOR will print out all changed lines, plus the last line examined.

#### Notes

1. Remember to TOP before making changes on the file as a whole.
2. If you end the command with a Carriage-Return, you can omit the closing delimiter.
3. You can specify the semi-colon (;) as a text character within the delimiters -- i.e. if you used "@" every place in your file where you wanted to use ";", then the command sequence TOP, CHANGE/@;/G9999 would change all the @'s to ;'s. (Make sure n is greater than the number of lines in your file.)

### The DELETE Command

The DELETE command deletes n lines, including the current line, and leaves the pointer at a null line where the last deleted line was. The null line will be maintained, in case you wish to insert a new line, until a new command moves the pointer away.

The format of the DELETE command is:

DELETE [n]

If n is not specified, the default value of 1 is used. n may be positive or negative, indicating deletion of the current line plus n-1 lines below or above the current line.

#### Note

Since n always includes the current line. The commands d, d1, and d-1 are equivalent.

```
print
'Twas brillig and the slithey toves
delete
print
.NULL.
```

#### Note

The DUNLOAD command, explained in the EDITOR REFERENCE section, permits you to move n lines to a new file, thus removing them from the work file without deleting them altogether. Although this technique accumulates files in your UFD, it can be useful if you are afraid of accidentally DELETEing large portions of your file.

### The INSERT Command

The INSERT command inserts newline following the current line; the inserted line then becomes the current line.

The format of the INSERT command is:

INSERT newline

```

print
'Twas brillig, and the slithey toves
insert Were doing their holiday shopping.
next-1
'Twas brillig, and the slithey toves
print 2
'Twas brillig, and the slithey toves
Were doing their holiday shopping.
Insert On Macy's! On Gimbel's! On Woolworth's and Kresge's!
print
On Macy's! On Gimbel's! On Woolworth's and Kresge's!

```

### The RETYPE Command

The RETYPE command deletes the current line and replaces it with text in string.

The format of the RETYPE command is:

```
RETYPE string
```

Remember: The first space after RETYPE separates the command word from the parameter; all further spaces are part of string.

```

print
Did gyre and gimbol in the wabe.
retype Did a wild watusi in the wabe.
print
Did a wild watusi in the wabe.

```

The string is terminated by either a semicolon (;) or a carriage-return (CR).

### Note

RETYPE followed immediately by a space and a Carriage-Return will act as a DELETE, erasing the current line and leaving the pointer at a blank line; RETYPE followed only by a CR will yield: BAD R

### ENDING AN EDITOR SESSION

Giving either the FILE command or the QUIT command tells EDITOR you are done editing the file. Each command has a specific use, which is explained below.

### The QUIT Command

The QUIT tells EDITOR you do not want to save the EDITOR work file, but instead, want to preserve the original and want to return to PRIMOS-level.

The format of the QUIT command is:

QUIT

If you have created/modified a file during the session, EDITOR will respond to a QUIT with:

FILE MODIFIED, OK TO QUIT?

This message asks whether EDITOR may throw away the work file.

A YES (or Y, YE, O, OK or Null line (CR)) response will QUIT you; you will get back an upper-case OK response, meaning you're at PRIMOS-level and your session was not saved. Any other response will provoke a PLEASE FILE (see FILE); doing a FILE, with or without a filename (depending on the circumstances), will automatically QUIT you, having saved your work.

If you did not create or modify a file, saying QUIT automatically returns you to PRIMOS.

```
OK, ed poem
GO
EDIT
print 2
.NULL.
'Twas brillig, and the slithy toves
quit
```

```
OK, ed poem
GO
EDIT
delete 3
quit
FILE MODIFIED, OK TO QUIT? yes
```

```
OK, ed poem
GO
EDIT
delete 3
quit
FILE MODIFIED, OK TO QUIT? no
PLEASE FILE
?
file short.poem

OK,
```



The FILE Command

The FILE command turns the EDITOR work file, (which is so far only a temporary) into a permanent file in your UFD and returns you from EDITOR to PRIMOS.

## WARNING

Since the work file has no existence outside of EDITOR, you must FILE if you want to save your work.

The format for the FILE command is:

FILE [filename]

The rules for using the FILE command are:

1. If you have been creating a new file, you must specify filename. (The error message FILENAME MUST BE SPECIFIED will occur if you don't.)

file

FILE NAME MUST BE SPECIFIED

?

file memol

OK,

You cannot have two files with the same name in the same UFD!! If you give a filename which already exists in your UFD, EDITOR will delete the old file by that name from your UFD, and put the EDITOR work file in its place.

2. The same warning holds true for old files. If you have been working on an old file, and you specify the old filename, or say FILE without any filename, your old copy will be deleted, and only your new version kept. Giving a new filename will keep both the old and new versions -- but be sure not to accidentally wipe out some other old file by using its name.
3. If you do not wish to save your work from a given session -- i.e., want to save your old version, if any -- type QUIT instead of FILE. If you have made any insertions or changes in your current file, EDITOR will inquire: FILE MODIFIED, OK TO QUIT? to double-check with you. A YES response QUITs you back to PRIMOS; NO provokes PLEASE FILE, at which point you give a FILE command.

quit  
 FILE MODIFIED, OK TO QUIT? no  
 PLEASE FILE  
 ?  
file new.memo

OK,

4. The rules for making filenames are:

- 1) Filenames can be up to thirty-two characters long.
- 2) The first character may be any character except a digit. However, starting with a letter is a good idea.
- 3) Filenames can contain only the following characters:

A through Z  
 0 through 9  
 & - \$ \* . \_ /

Characters NOT permitted in filenames include:

Imbedded blanks  
 Special characters like " ? ' @ ;

- 4) Upper and lower case letters are treated as upper case by PRIMOS. (Letters inputted in lower-case will be converted to upper-case.)

NEWFILE  
 Todays-Prices  
 Highs&Lows  
 \$Monthly.Report  
 R34587  
 A-Tale-Of-Two-Cities

Note

The FILE command can also be used to make copies of any file, simply by typing ED plus filename and FILEing the copied work file immediately with a new filename, like so:

OK, ed memol  
 GO  
 EDIT  
file spare-memol

OK,

## MISCELLANEOUS INFORMATION CONCERNING EDITOR

Tabs

Do NOT try to input tabs with your TAB key! EDITOR will not understand them. To signal a tab, use the Backslash (\) character. Typing this line of input

use\them\tabs\wisely, buddy

would be interpreted as

use    them    tabs                      wisely, buoöy

EDITOR has pre-set tab stops at columns 6, 12, and 30. To change these and/or add more tab stops, see the page on the TAB command in the EDITOR REFERENCE SECTION.

Entering the Double-Quote and Question-Mark in Your Text

As you know, the double-quote (") and question-mark (?) have special meanings in EDITOR. Every " erases the previous character; a ? kills (erases) the entire line.

You can enter a " or ? as actual characters in your text input by preceding it with an up-arrow (^). The table below shows the results of various combinations of ^, " and ?.

<u>You type</u>	<u>Result</u>
"	Erase previous character
^"	Enters a "
^""	None (Enters ", then erases it)
?	Kill line
^?	Enters a ?
^??	Kill entire line, including text ?
^?"	Erases the entered ?
^"?	Kills entire line
^?^"	Enters ?"

Here's a few sample sessions in EDITOR.

```
ed
GO
INPUT
Jerry--;;Just a quick note...we're releasing
the news of our new product line
on Tuesday, and I wanted you to
have a ?have an advance peek ".;
EDIT
p
have an advance peek.

INPUT
;Check out the smart cars, in
particular, and dig those pet moon
rocks--they're the most! (Say
the word and I'll get you one)
\\Dig you later,;\\Delos D. Harriman

EDIT
top,print4
.NULL.
Jerry--

Just a quick note...we're releasing
locate Tuesday
on Tuesday, and I wanted you to
c/Tuesday/next Tuesday/
on next Tuesday, and I wanted you to
c/sa/s
on next Tuesday, and I wanted you to
find the news
BOTTOM
top,find the news
the news of our new product line
a (Ta-DA!)
the news of our new product line(Ta-DA!)
file jerry

OK, ed jerry
GO
EDIT
b
Insert P>S>?i P.S. Congrats on the new 'droid!ddh
file

OK, slist jerry
GO
Jerry--
```

```
Just a quick note...we're releasing
the news of our new product line(Ta-DA!)
on next Tuesday, and I wanted you to
have an advance peek.
```

```
Check out the smart cars, in
particular, and dig those pet moon
rocks--they're the most! (Say
the word and I'll get you one)
Dig you later,
Delos D. Harriman
P.S. Congrats on the new 'droid!ddh

OK,
```

Give ED command to create  
a new file.  
EDITOR begins in INPUT Mode,  
Start typing file. Note that  
you skip lines by typing  
semicolons.

Typing a Carriage-Return after  
a semicolon put you into  
EDIT Mode. Typing CR again  
returns you to INPUT Mode,  
below the most recent line,  
in this case.

Continue to input file. Note  
use of the backslash to tab.

Switch back to EDIT Mode to  
check your work.  
Remember: This .NULL. is not  
actually part of your file!

Find the word "Tuesday" which  
appears somewhere on a line.  
Change "Tuesday" to "next Tuesday"  
Observe spelling error.  
Correct it.

Find line beginning with "the news"  
It's above where you started from.  
Go to TOP and try again.

Add something to this line.

File the work under JERRY

Give ED command plus JERRY to  
edit this file.

You're in EDIT Mode.

Go to BOTTOM

Insert a PS line below the last  
line. Note use of Kill Character  
(?) to correct typing error.  
Having filed the changed file,  
use the SLIST command to check  
what you've got.

Here's a summary of what you've learned so far:

#### PRIMOS

<u>Command Format</u>	<u>Meaning</u>
<u>ED</u>	Enter EDITOR in Input Mode to start a new file.
<u>ED</u> filename	Enter EDITOR in Edit Mode to work on <u>filename</u> .
<u>LOGIN</u> ufd-name [password]	Gets you on computer, gives you a work space, and attaches you to your UFD.
<u>LISTF</u>	Lists names of all files in your UFD.
<u>LOGOUT</u>	Tells system the session is over.

#### EDITOR

<u>Command Format</u>	<u>Meaning</u>
<u>APPEND</u>	Append <u>string</u> to end of current line.
<u>BOTTOM</u>	Move pointer to null line at bottom of work file.
<u>CHANGE</u> /string-1/string-2/[G] [n]	Change <u>string-1</u> to <u>string-2</u> in next <u>n</u> lines, either first occurrence, or Generally.
<u>DELETE</u> [n]	Delete <u>n</u> lines, beginning with the current line.
<u>FILE</u>	Save work file into current UFD under old <u>filename</u> .
<u>FILE</u> filename	Save work file into current UFD as sub-UFD <u>filename</u> and delete previous file named <u>filename</u> from UFD, if there was one.
<u>FIND</u> string <u>FIND</u> (n) string	Move the pointer to the first line below the current line which contains <u>string</u> , beginning in column 1 or <u>n</u> .
<u>INSERT</u> newline	Insert <u>newline</u> below the current line, and make <u>newline</u> the new current line.

<u>LOCATE</u> string	Move the pointer to the first line below the current line which contains <u>string</u> anywhere on it.
<u>POINT</u> n	Move the pointer to line <u>n</u> .
<u>PRINT</u> [n]	Print n lines, beginning with the current line, and move the pointer to the last line printed.
<u>QUIT</u>	Throw away work file and leave EDITOR.
<u>RETYPE</u> string	Delete current line and insert <u>string</u> in its place.
<u>TOP</u>	Move pointer to null line at top of work file.
<u>WHERE</u>	Print current line number.

#### General Information About Editor

EDITOR works on a copy of the original file, called the work file.

The Carriage-Return (CR) inputs a line of typing.

The double-quote (") erases the previous character.

The question-mark (?) erases the entire line.

The backslash (\) is the tab character; EDITOR has pre-set tabs in columns 6, 12, and 30.

The pointer indicates the current line.

All commands work on the current line.

#### EDITOR'S OTHER COMMANDS

Besides the commands you have just learned, EDITOR has additional commands. These are briefly described here, along with their uses. For more information about each command, see the EDITOR REFERENCE SECTION.

#### The File Loading & Unloading Commands

These commands allow you to transfer lines from the work file into a new file, and insert other files from your UFD into the work file.

<u>LOAD</u> filename	Inserts (Loads) a copy of <u>filename</u> into EDITOR's work file below the current line, and repositions pointer to bottom of LOAded file.
----------------------	---

*add 2*

UNLOAD filename [n] Copies (unloads) n lines into filename from EDITOR work file, beginning at current line.

UNLOAD filename TO string Copies (unloads) lines into filename from EDITOR work file beginning at current line, until a line containing string is found.

DUNLOAD filename [n] Deletes n lines from work file after copying (unloading) them into filename beginning with current line.

DUNLOAD filename TO string Deletes every line in the work file between the current line and the first line containing string, after copying these lines into filename.

Line-changing Commands

These commands affect the contents of specified lines.

DELETE TO string Deletes from current line to first line containing string.

MODIFY/string-1/string-2/ [G] [n] Superimposes string-2 on string-1 for n lines beginning with current-line; either first occurrence on a line, or G (for Generally).

GMODIFY A powerful but complicated command which permits very specific line changes. See the EDITOR REFERENCE SECTION.

MOVE buffer-1 { buffer-1 /string/ } Moves one line of text from buffer-2 to buffer-1. Buffer names are STRA, STRB, INLIN, and EDLIN.

Control Commands

These commands affect command execution.

\* [n] Repeats preceding command n times; or if n is omitted, until bottom of work file is reached.

XEQ buffer Executes contents of buffer.

PAUSE Returns you to PRIMOS without wiping out EDITOR work file. Re-start via START.

The Symbol-changing Commands

These commands allow you to check and change the values of EDITOR's reserved characters.

PSYMBOL

Prints list of current symbols and functions. The defaults

```

KILL      ?
ERASE     "
BLANKS    #
WILD      !
TAB       \
ESCAPE    ~
CPROMP    $
DPROMP    &
COUNT   @
SEMICO    ;

```

SYMBOL name character

Changes current value of the reserved character name to character.

ERASE character

Makes character new Erase character.

KILL character

Makes character new Kill character.

The Mode Commands

The mode commands allow you to turn on and off various EDITOR features.

VERIFY

Activates display of verification lines (prints current line after certain commands). Default.

BRIEF

Suppresses verification responses.

MODE PROMPT

Causes EDITOR to print Input and Edit Mode prompts.

MODE NPROMPT

Stops printing of Input and Edit Mode Prompts. Default.

MODE NUMBER

Turns on display of line numbers. These numbers are not part of the actual work file.

MODE NNUMBER

Turns off display of line numbers. Default.

MODE COLUMN

Causes EDITOR to display column numbers whenever you enter Input Mode.



<u>MODE NCOLUMN</u>	Turns off column number display. Default.
<u>MODE COUNT</u>	Turns on a counter.
<u>MODE NCOUNT</u>	Turns off a counter.
<u>MODE</u> $\left\{ \begin{array}{l} \underline{\text{PRALL}} \\ \underline{\text{PRUPPER}} \\ \underline{\text{PRLOWER}} \end{array} \right\}$	The case.modes tell terminals which print only upper case letters to flag upper and lower case letters with the signals ↑U and ↑L.

### Value-Setting Commands

<u>LINESZ</u> n	Change maximum line width to <u>n</u> characters.
<u>PTABSET</u> tab-1...tab-8	Informs EDITOR of physical device tab settings. Do <u>not</u> confuse with TAB command!
<u>TABSET</u> tab-1...tab-8	Sets up to eight logical tab stops obtained by tab symbol.

### Input/Output Commands

These commands control input and output modes.

<u>INPUT</u> $\left\{ \begin{array}{l} \underline{\text{ASR}} \\ \underline{\text{PTR}} \\ \underline{\text{TTY}} \end{array} \right\}$	Read input from specified device. The terminal (TTY) is the default.
<u>PUNCH</u> $\left\{ \begin{array}{l} (\underline{\text{ASR}}) \\ (\underline{\text{PTP}}) \end{array} \right\}$ n	Punch <u>n</u> lines from work file on high or low-speed paper tape.

You now know enough to log into the system, create new files with EDITOR, and edit them. Section 4 gives you more PRIMOS-level information on dealing with your files. The EDITOR Reference Section contains complete information on all EDITOR's commands. At this point, you should be able to read through this section, or whatever parts of it you want, and have no trouble learning the additional commands.

## SECTION 4

## MORE PRIMOS

You've now learned how to log in to your UFD, make and edit files, get a list of these files, and log out, using the LOGIN, ED, LISTF and LOGOUT commands. There are several other PRIMOS-level commands which can be useful. These are the ATTACH, CREATE, SPOOL, SLIST, CNAME, and DELETE commands.

## USING OTHER UFDs: THE ATTACH COMMAND

In Section 2, we mentioned that once you have logged in to your UFD via the LOGIN command, you can not only use the files in your own UFD, but also those in other UFDs.

However, in order to use these other files, you must first get access to the right "drawer".

The ATTACH command tells PRIMOS to "open a new drawer" to you. Since you can only work from one UFD at a time, PRIMOS will "close your drawer" and then give you access to this new UFD.

The format of the ATTACH command is:

```
ATTACH ufd-name [ password ]
```

When you give an ATTACH command, PRIMOS gives you access to this new UFD. If at this point you give the LISTF command, PRIMOS gives you a list of the files in this new UFD; if you edit a new file, when you give the FILE command PRIMOS will put this new file into this UFD which you are currently ATTACHED to.

```
OK, attach staff  
OK, listf
```

```
UFD=STAFF 3 0
```

```
FRANK  ALBERT  ROSEMARY      MARTHA
```

```
OK,
```

The UFD to which you are currently attached is known as your current UFD.

You may give the LOGOUT command while attached to another UFD than your own; PRIMOS will still "sign you out" correctly.

#### MAKING SUB-UFDs: USING THE CREATE COMMAND

The CREATE command allows you to make subdivisions in your UFD called sub-UFDs. In order to explain the CREATE command, we must first explain the concept of sub-UFDs.

##### What is a SUB-UFD?

As you recall, your User File Directory is a list of Files. A Sub-UFD is a selection of those Files which have been grouped together. This grouping has a name called a sub-UFD-name. Once you have logged in to your UFD, you can ATTACH to any sub-UFD within it. This sub-UFD would then be your CURRENT DIRECTORY -- giving the LISTF command would get you a list of only those Files within this sub-UFD.

You can make sub-UFDs within sub-UFDs.

So; your UFD can contain FILES and sub-UFDs. A sub-UFD can contain FILES and sub-UFDs.

##### The CREATE Command

The command which lets you define and name a sub-UFD is the CREATE command. Whenever you give a CREATE command, PRIMOS creates a sub-UFD within whatever UFD or sub-UFD you are currently in -- i.e., if you are working in your UFD, PRIMOS creates a sub-UFD within the UFD; if you are working in a sub-UFD of a sub-UFD of your UFD, PRIMOS will create a sub-UFD within that sub-UFD of a sub-UFD.

The format of the CREATE command is:

CREATE sub-ufd-name

PRIMOS will not let you have two files or sub-UFDs with the same name in your current UFD. If you attempt to CREATE a sub-UFD with a name already in use, you will get the message: ALREADY EXISTS.

OK, attach sales  
 OK, listf

UFD=SALES 3 0

.NULL.

OK, create east.coast  
 OK, create west.coast  
 OK, create east.coast  
 ALREADY EXISTS.  
 ER! listf

UFD=SALES 3 0

EAST.COAST        WEST.COAST

OK,

### Using These Sub-UFDs

In order to work within a sub-UFD, you must first ATTACH to it. The format of the ATTACH command for sub-UFDs is:

ATTACH sub-ufd-name 1/2

You must include the one-slash-two (1/2) in order to ATTACH to a sub-UFD! This group of characters identifies sub-ufd-name as the name of a sub-UFD in your current UFD. If you omit the one-slash-two, PRIMOS will give the error message:

NOT FOUND

When you have attached to a sub-UFD, it becomes your current UFD.

You have now learned two uses for the ATTACH command:

1. Attaching to a UFD.
2. Attaching to a sub-UFD within a UFD or sub-UFD.

These are the only two ways to use the ATTACH command. In order to attach to any other sub-UFD than one in your current UFD (remember that your current UFD may be a sub-UFD!) you must first ATTACH to your UFD, and then ATTACH down, sub-UFD by sub-UFD.

Suppose you have a UFD called SALES in which you want to create the sub-UFDs East.Coast and West.Coast. You also want both East.Coast and West.Coast to contain sub-UFDs names North, Central and South. You also want the sub-UFD North in the sub-UFD East.Coast to contain three further divisions: MAIN, BRANCH1 and BRANCH2.

Here's how you would do it:

```

attach sales
OK, listf

UFD=SALES 3 0

EAST.COAST      WEST.COAST

OK, attach east.coast 1/2
OK, create north
OK, create south
OK, create central
OK, listf

UFD=EAST.COAST 3 0

NORTH  SOUTH  CENTRAL

OK, attach north 1/2
create main
OK, create branch1
OK, create branch2
OK, attach sales
OK, attach east.coast 1/2
OK, attach north 1/2
OK, listf

UFD=NORTH 3 0

MAIN      BRANCH1  BRANCH2

```

#### LOOKING AT YOUR FILES USING THE SLIST AND SPOOL COMMANDS

The SLIST and SPOOL commands allow you to look at a file in your current UFD (which may be a sub-UFD). SLIST command displays a file on your terminal; the SPOOL command has a file printed out on the line printer.

##### The SLIST Command

The SLIST command displays a file on your terminal.

The format of the SLIST command is:

SLIST filename

OK, slist poem

GO

'Twas brillig, and the slithy toves  
Did gyre and gimbol in the wabe.  
All mimsey were the borogoves  
And the mome raths outgrabe.

"Beware the Jabberwock, my son!  
The jaws that bite, the claws that snatch--  
Beware the Jub-Jub bird, my son,  
And shun the frumious Bandersnatch!"

He took his vorpal sword in hand,  
Long time the manxome foe he sought;  
And rested he by the tum-tum tree  
And stood awhile in thought.

OK,

This command is useful on CRT terminals only for very short files; longer files will zip past you faster than you can read them. However, if you have a hard-copy terminal nearby, you can get a copy of any file by using this command. (You'll have to log in at this terminal, of course).

#### The SPOOL Command

You may prefer to use the SPOOL command, particularly if the file is long. This way you can avoid tying up a terminal while printing a long file.

The SPOOL command has a hard copy of your file printed out on the line printer in the computer room. You will have to pick up the print-out when it is ready.

The format of the SPOOL command is:

SPOOL filename

When you give this command, PRIMOS makes a note of the filename in the Spool Queue List for the line printer, and displays the message:

YOUR SPOOL FILE IS PRTxxx

Where xxx is a 3-digit number that identifies your file in the Spool Queue List. The reason there is a list, rather than just having each file spooled out as the request comes, is that some requests are very long - hundreds of pages. PRIMOS spools out the shorter files as soon as possible, rather than make the users wait while the long files are printed.

The word (SHORT) or (LONG) which follows the SPOOL message tells you which of these categories your file is in. You can check on the status of your SPOOL request -- get a list of those files waiting in the Spool Queue, and see where yours is -- by giving the command:

SPOOL -LIST

For example:

OK, spool poem

GO

YOUR SPOOL FILE IS PRT013 (LONG) REV 14.00

OK, spool -list

GO

USER	FILE	DATE/TIME	OPTS	SIZE	NAME	FORM
COSMO	PRT005	11/09 10:03	S	5	RIN168	WHITE
MARTHA	PRT007	11/09 15:34	S	3	RIN172	WHITE
HAMPSO	PRT009	11/11 10:25	S	6	PAGTUR	
LAWLER	PRT010	11/11 10:26	S	9	L AM9600	
RANDI	PRT011	11/11 10:26	S	1	WKINFO	
MORRIS	PRT012	11/11 10:27	S	3	L CMP\$SR	
ALICE	PRT013	11/11 10:28	L	17	POEM	

If for some reason you decide you do not want your file to be spooled -- for example, you gave the wrong filename, or discovered that you still have to correct or change something -- you may cancel your spool request with the command:

SPOOL -CANCEL PRTxxx

where xxx is the number of your Spool File.

For example:

OK, spool -cancel prt013

GO

PRT013 CANCELLED.

OK,



Notes

1. If PRIMOS has begun SPOOLing your file, you will not be able to cancel a request, and will instead get the message:

CAN'T CANCEL REQUEST - FILE IS OPEN OR CURRENTLY  
PRINTING

2. If your file is done SPOOLing, or you gave a non-existent PRT number, you will get the message:

PRTxxx NOT IN QUEUE

3. If you give the name of the file instead of the PRT number, you will get the message:

BAD PRINT FILE NAME

The SPOOL command is useful for getting print-outs of files without tying up your printing terminals, or if you do not have a printing terminal. If you want to inspect a file on a CRT terminal, you can use the ED command and check blocks of lines with successive PRINTs.

RENAMING AND DELETING FILES AND SUB-UFDs

The CNAME and DELETE commands allow you to rename and delete both files and sub-UFDs.

The CNAME Command

The CNAME command allows you to change the name of a file or a sub-UFD.

The format of the CNAME command is:

CNAME old-filename new-filename

If you give a name for new-filename which already exists in your current UFD, PRIMOS displays the message:

ALREADY EXISTS

```

OK, cname west.coast california
OK, listf

UFD=SALES  3 0

EAST.COAST      CALIFORNIA

OK, cname east.coast california
ALREADY EXISTS.
ER! cname california west.coast
OK, listf

UFD=SALES  3 0

EAST.COAST      WEST.COAST

OK,

```

If you misspell the name of the old filename, PRIMOS displays the message:

```

NOT FOUND
ER!

```

### The DELETE Command

The DELETE command allows you to delete files and empty sub-UFDs.

The format of the DELETE command is:

```

DELETE { filename
          sub-ufd-name }

```

You cannot delete a sub-UFD which contains files. To delete such a sub-UFD, you must first delete all the files in it. This procedure prevents you from accidentally wiping out a large quantity of work in one unintended command; if you try to delete a sub-UFD which still contains files, you will get the message:

```

DIRECTORY NOT EMPTY

```

OK, listf

UFD=SALES 3 0

EAST.COAST WEST.COAST

OK, delete east.coast

DIRECTORY NOT EMPTY.

ER! attach east.coast 1/2

OK, listf

UFD=EAST.COAST 3 0

NORTH SOUTH CENTRAL

OK, delete north

DIRECTORY NOT EMPTY.

ER! delete south

OK, delete central

OK, listf

UFD=EAST.COAST 3 0

NORTH

OK,

With these commands, you should be able to arrange and use your files to your best advantage, and get rid of old or unwanted files as you go along.

login staff

STAFF (15) LOGGED IN AT 10'36 110579

listf

UFD=STAFF 3 0

FRANK MARTHA ROSEMARY ALBERT

OK, attach sales

OK, listf

UFD=SALES 3 0

.NULL.

OK, create east.coast

OK, create west.coast

OK, create east.coast

ALREADY EXISTS.

ER! listf

UFD=SALES 3 0

EAST.COAST WEST.COAST

OK, attach east.coast 1/2

OK, create north

OK, create south

OK, create central

OK, listf

UFD=EAST.COAST 3 0

NORTH SOUTH CENTRAL

OK, attach north 1/2

OK, create main

OK, create branch1

OK, create branch3

OK, delete branch3

OK, create brunch2

OK, cname brunch2 brunch2

OK, attach sales

OK, attach east.coast 1/2

OK, attach north 1/2

OK, listf

UFD=NORTH 3 0

MAIN BRANCH1 BRANCH2

OK, attach staff

OK, listf

UFD=STAFF 3 0

Figure 4-1. Advanced PRIMOS Session

FRANK    ALBERT   ROSEMARY            MARTHA

OK, attach albert

OK, listf

UFD=ALBERT   3 0

MEM01    UPDATES       \$PHONE.LIST

OK, slist updates

GO

Albert's Update List as of Nov 12, 1984

Get wide-beam sample

TALK WITH MARKETING RE BROCHURES!!

Call Sid

Exchange samples with Galactic

Re-order for Jerry

OK, ed mem01

GO

EDIT

change/NAME/Jerry/10

Dear Jerry

file jerry

OK, spool jerry

GO

YOUR SPOOL FILE IS PRT013   (LONG)   REV 14.00

GO

OK, spool -list

GO

USER	FILE	DATE/TIME	OPTS	SIZE	NAME	FORM	DEFER
COSMO	PRT005	11/09 10:03	S	5	RTN168	WHITE	
MARTHA	PRT007	11/09 15:34	S	3	RTN172	WHITE	
HAMPSO	PRT009	11/11 10:25	S	6	PAGTUR		
LAWLER	PRT010	11/11 10:26	S	9	L AM9600		
RANDI	PRT011	11/11 10:26	S	1	WKINFO		
MORRIS	PRT012	11/11 10:27	S	3	L CMP\$SR		
STAFF	PRT013	11/11 10:28	L	17	JERRY		

OK, spool -cancel prt013

GO

PRT013 CANCELLED.

OK, logout

STAFF    (15) LOGGED OUT AT 11'28 110579

TIME USED= 0'12   0'06   0'032

OK,

## SECTION 5

## THE ESSENTIALS OF RUNOFF

## INTRODUCTION

RUNOFF is Prime's text-processing system. It turns a file like this:

```
.paragraph 5 1
"My dear Fortunato, you are luckily met. How
remarkably well you are looking
today. But I have received a pipe of what passes
for Amontillado, and I have my
doubts."
.paragraph
"How?" said he. "Amontillado? A pipe?
Impossible! And in the
middle of the carnival!"
```

into a file like this:

```
"My dear Fortunato, you are luckily met.
How remarkably well you are looking today.
But I have received a pipe of what passes for
Amontillado, and I have my doubts."

"How?" said he. "Amontillado? A pipe?
Impossible! And in the middle of the
carnival!"
```

Using EDITOR, you create a text file containing RUNOFF commands specifying how you want RUNOFF to format the text output. Using only a handful of commands, you can paginate, right-justify, paragraph, indent, do headers and footers, and more.

RUNOFF allows you to change output formats and see them without your having to retype any text whatsoever -- all you do is change a command or two. RUNOFF can generate indexes, tables of contents, do decimalized headings; write a series of form letters inserting a different name on each, and combine separate files of text and tables into one master output file.

Unlike EDITOR, where you must explicitly do everything that you want done, RUNOFF does a lot of work for you, unless you tell it not to. RUNOFF's defaults include setting up a standard page size, right-justifying your text, paragraphing a standard way. This means that you don't have to know a lot about RUNOFF in order to use it.

Using RUNOFF is a two-part process, and for this reason, RUNOFF will be explained in a different manner from EDITOR:

- First, a basic explanation of how to create a RUNOFF source input file, using EDITOR.
- Next, how to turn this source file into a processed output file.

This is enough to let you get the hang of it. Then, in Section 6, MORE RUNOFF, you can learn additional commands to aid you in setting up your formats and outputting your files. As with EDITOR, the RUNOFF REFERENCE SECTION contains a complete alphabetized list of the RUNOFF commands, with full explanations of how to use them, and the RUNOFF Summary in Appendix A supplies an easy-to-use table of RUNOFF command formats.

#### CREATING THE SOURCE FILE

In order to process text, RUNOFF requires the following two items:

1. Text to process
2. Commands specifying how to process this text.

You have already learned how to make a file of text using EDITOR.

Now you will learn how to give the commands that tell RUNOFF what to do with this file.

These commands are called RUNOFF commands. You put RUNOFF commands directly into your text file, using EDITOR, transforming it into a RUNOFF source file.

When you have finished creating your RUNOFF source file (and FILED it, of course), you are ready to give the RUNOFF command to PRIMOS. The RUNOFF command tells PRIMOS to start the RUNOFF program, and gives it the indicated file to be processed. RUNOFF then reads through this source file, and processes the text according to the commands; the results are placed into the RUNOFF output file. This output file contains your formatted text, and is available for SLISTing, SPOOLing, and/or EDITing.

This process is illustrated in Figure 5-1.

#### RUNOFF COMMANDS

RUNOFF commands control the way that RUNOFF processes the text in the source file into the output file. You create a file containing text and RUNOFF commands by using EDITOR.

Before we explain the basic RUNOFF commands, here's a brief discussion of RUNOFF's command conventions.

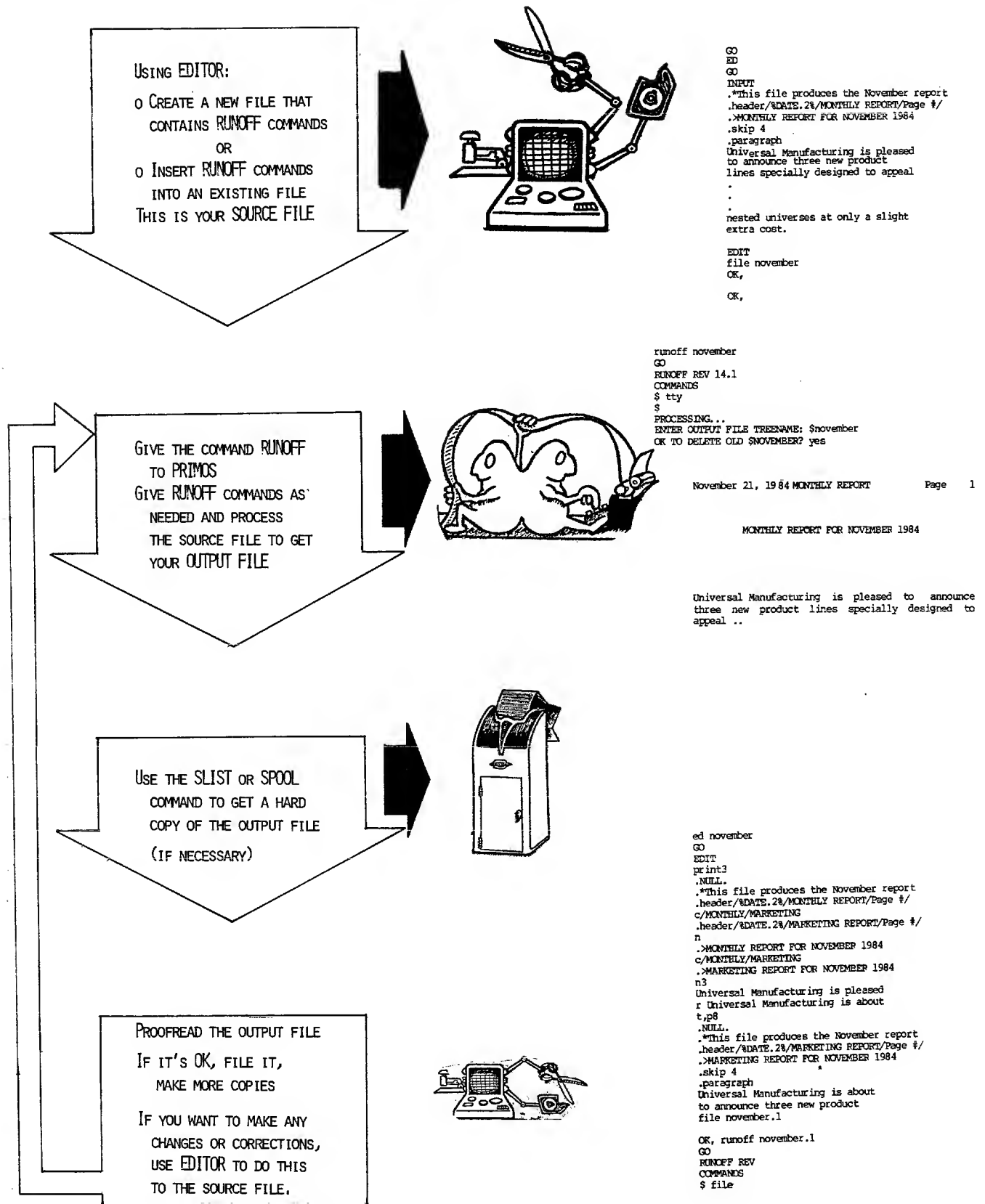


Figure 5-1. Sequence of Using RUNOFF



## RUNOFF COMMAND CONVENTIONS

A RUNOFF command consists of a period (.) followed by a command word, and possibly one or more parameters, like so:

.COMMAND parameter

The Period: The period (.) signals that this line in the source file contains a command, which is to be obeyed, as opposed to text, which is to be processed.

Command Words: A command word is a word which specifies an action. In this manual, the underlining of certain letters in a command word indicate the minimum acceptable abbreviation of the word which RUNOFF will understand. For example, if the format indicated:

.PARAGRAPH

then any of the following would be acceptable in your source file:

.P  
.PAR  
.PARAGRAPH

You may input command words in either upper or lower case (or a combination).

Parameters: In RUNOFF, a parameter is either the name of a file, a number specifying a line, column, page number, number of lines or columns, or a piece of text, depending on the command. Numerical parameters are represented in the formats by:

the letter i for a general number, usually of pages  
the letter m for a number of spaces  
the letter n for a number of lines.

Some commands do not use parameters, some require them, and in others they are optional. In this manual, parameters which are enclosed in brackets - e.g., [filename] - are optional. If you omit them, RUNOFF will use the appropriate default value.

## TYPES OF COMMANDS

RUNOFF commands fall into the following categories:

- Page-Formatting - control the physical appearance of pages.
- Line-Formatting - control the processing of words and lines of the input text into lines on the formatted pages.
- Output - Specify way to process the output file.

- Characters & Symbols - define special characters and meanings.
- Features - indexing, tables of contents, decimalization.

Because of RUNOFF's built-in defaults, a text file containing no RUNOFF commands whatsoever would be processed onto 8½ by 11 inch pages as right-justified text. With the addition of only a few commands, you can do tabbing, paragraphing, indentation, headers and footers, and other basic formatting.

#### PAGE-FORMATTING COMMANDS

Page-formatting commands define the desired margins and physical size of the page, plus headers, footers and page numbering.

##### Margins and Page Size

The default page size and margins are illustrated in Figure 5-2; if you do not specify anything, you get this 8½ by 11 inch page, containing 54 lines of 71 character spaces each. The page margins are 7 spaces on the left and right sides, 7 lines on top, and 5 lines on the bottom. (The commands to change these defaults are described in Section 6 -- MORE RUNOFF.)

##### Headers & Footers

You can set up one-line headers and/or footers positioned in the middle of the top and bottom margins respectively, by the commands:

.HEADER/left-text/center-text/right-text/

.FOOTER/left-text/center-text/right-text/

The headers and footers will start on the first new page after command. Headers and footers may contain left-justified, centered, and/or right-justified portions. All of the four delimiters must be in each HEADER or FOOTER command, even if a given text portion is left blank.

The slash (/) is used as the delimiter in the description of all Header and Footer formats, but you may use any character that does not appear in the text portions as the delimiter. This is demonstrated by the examples in the RUNOFF REFERENCE SECTION.

The placement of Headers and Footers is indicated on the default page illustration, Figure 5-2.

##### Page Number

You can insert the current page number into the header or footer by using the Pound Sign (#) in the format. During processing, RUNOFF will replace each # with the current page number. For example, given this command:

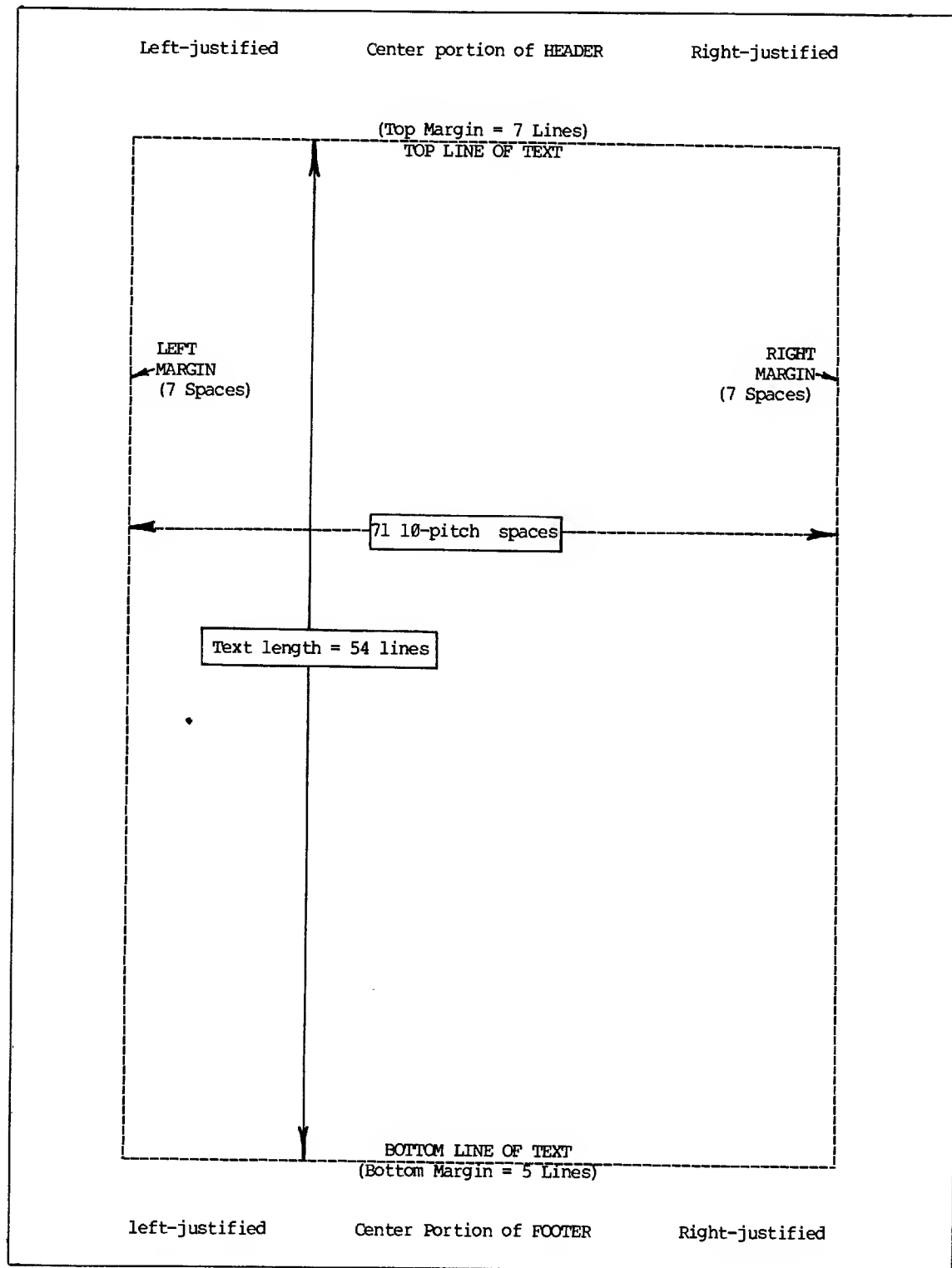


Figure 5-2. RUNOFF Default Page Specifications

.HEADER/UNIVERSAL MFG CORP/NEW PRODUCTS/Page #/

the twelfth page of the output file would have this header:

UNIVERSAL MFG CORP

NEW PRODUCTS

Page 12

You only need to specify a header or footer once. RUNOFF will print them on all subsequent pages and update the page number automatically. These command(s) must appear before the first text line of a page in order to be printed on that page.

#### LINE-FORMATTING COMMANDS

Now you've got a page size, plus headers and/or footers which can contain the page number. The next thing to do is to specify the general way you want RUNOFF to process each word or line of text from the source file to the output file. This is done with Line-Formatting Commands.

There are two types of line-formatting commands -- General and Local. General line-formatting commands specify a manner of processing which is to stay in effect until you specify otherwise. Local line-formatting commands affect only the text at the point where the command occurs.

#### General Line-Formatting Commands

General line-formatting commands, when given, stay in effect until you explicitly change them. RUNOFF has five commands of this type -- SPACE, TAB, and the three modes FILL, ADJUST, and NFILL.

The modes define how RUNOFF will move text from the source file to the output file. You can choose between:

- FILL Mode - Fill each output line with words from the input file.
- ADJUST Mode - Right-justify each filled line by adjusting inter-word spacing.
- NFILL Mode - Do not fill; transfer text line by line.

The commands for these modes are:

```
.FILL
.ADJUST (Default)
.NFILL
```

Since ADJUST Mode is the default, you will automatically get right-justified output, unless you specify otherwise.

FILL Mode: FILL Mode produces a "ragged right" margin. Both ADJUST and FILL Modes ignore all tab characters unless the input line begins with one; they also ignore all extra spaces between words. FILL puts exactly one space between words, unless the word is followed by either a period (.), semi-colon (;), exclamation point (!), question mark (?), or a right parenthesis which was preceded by any of these except the semi-colon, in which case you will get two spaces.

#### Note

The last line of a paragraph is never FILLed.

ADJUST Mode: In ADJUST Mode, once a line is filled, RUNOFF distributes extra spaces across the line, wherever there are spaces, until the line touches both left and right margins. This is also called "right-justified".

NFILL Mode: In NFILL Mode, RUNOFF transfers each line of text, exactly as it appears in the source file, to the output file. All tabs are obeyed; all multi-column spaces are maintained. NFILL Mode is therefore useful in processing tables.

Here is how RUNOFF processes a sample of a source file, in each of the three modes:

```
.paragraph 5 1
"My dear Fortunato, you are luckily met. How
remarkably well you are looking
today. But I have received a pipe of what passes
for Amontillado, and I have my
doubts."
.paragraph
"How?" said he. "Amontillado? A pipe?
Impossible! And in the
middle of the carnival!"
```

In ADJUST mode:

```
"My dear Fortunato, you are luckily met.
How remarkably well you are looking today.
But I have received a pipe of what passes for
Amontillado, and I have my doubts."
```

```
"How?" said he. "Amontillado? A pipe?
Impossible! And in the middle of the
carnival!"
```

In FILL mode:

"My dear Fortunato, you are luckily met.  
How remarkably well you are looking today.  
But I have received a pipe of what passes for  
Amontillado, and I have my doubts."

"How?" said he. "Amontillado? A pipe?  
Impossible! And in the middle of the  
carnival!"

In NFILL mode:

"My dear Fortunato, you are luckily met. How  
remarkably well you are looking  
today. But I have received a pipe of what passes  
for Amontillado, and I have my  
doubts."

"How?" said he. "Amontillado? A pipe?  
Impossible! And in the  
middle of the carnival!"

RUNOFF will always use whichever of these three modes has been most recently specified. If you do not specify a mode, the default of ADJUST is used.

The SPACE Command: The other two General Line-Formatting commands are SPACE and TAB. The SPACE command defines the spacing between lines of print -- single-space, double-space, triple-space, and so forth. The format is:

.SPACE [n]

If n is omitted or 0, the default value of 1 (single-spacing) is used.

The TAB Command: The TAB command defines the character and tab stop columns for a file. As we mentioned earlier, RUNOFF and EDITOR do tabbing differently. EDITOR inserted the appropriate amount of spaces into your file whenever you typed the current EDITOR tab symbol. But, when processing in FILL or ADJUST Mode, RUNOFF would ignore these spaces. You must explicitly tab in RUNOFF. RUNOFF requires you to define your tab character explicitly before you may use it. The format of the TAB command is:

.TAB character tab-1 tab-2...tab-8

The eight tab stops must be column numbers, with the higher ones to the right.

RUNOFF processes tab symbols:

- In NFILL Mode
- If an input line begins with a tab symbol

In either case, all characters on an input line are processed. In FILL and ADJUST Modes, tab characters are not processed unless the input line begins with one.

In the following example, the @ character is defined as the Tab Symbol, and the tab stops are set in columns 7, 27, and 47.

```
.tab @ 7 27 47
.nfill
@In NFILL MODE@All @tab
characters are@recognized
.adjust
@In ADJUST @and FILL@Modes
tab @characters@ are only recognized
if @one begins @ a line
```

yields

In NFILL MODE	All	tab
characters are	recognized	
In ADJUST	and FILL	Modes
tab @characters@	are only recognized	if @one begins
@ a line		

Notice how in ADJUST and FILL Modes, the @ character is not interpreted as a tab character because it did not begin a line in the Source File.

Tabs are primarily useful for formatting tables.

Local Line-Formatting Commands: Local line-formatting commands have an effect only when you explicitly give them, although they may also set a numerical value for a parameter which will be remembered.

Local line-formatting commands insert either a text line or a specified number of blank lines.

The commands which insert text lines, and their meanings, are:

- .>text      The right-angle bracket (>) signals RUNOFF to center text on the line. The command is particularly useful for titles and captions.
- .+text      The plus-sign tells RUNOFF to insert this line verbatim -- i.e. exactly as it appears here. This command permits you to override FILL or ADJUST Mode for a single line.

./left-text/center-text/right-text/

The four delimiters tell RUNOFF to apportion the pieces of text as left-justified, centered, and right-justified portions of text. You may omit any of the text portions, but must still give all four delimiters. The slash (/) is the only permissible delimiter here. This means the / may not be used as a text character.

.\*comment The asterisk (\*) tells RUNOFF to ignore this line completely. This permits you to insert comments within your source file, to document what you have done at a given point.

Here's an example of these four commands:

The source file:

```
.*This file contains RUNOFF line-insertion commands.
>EXAMPLES OF LINE INSERTION
.*This line inserted literally.
./Left piece/I'm the Middle/Right On/
```

would process to:

	EXAMPLES OF LINE INSERTION		
This	line inserted	literally.	
Left piece	I'm the Middle		Right On

The commands which insert blank lines are BREAK, SKIP, PARAGRAPH and EJECT.

The BREAK command tells RUNOFF to stop filling the current output line at once. The line is ended, and not adjusted, no matter what mode you are in. The next text in the source file will be put onto a new output line. This action of breaking off the transferring of text to a line is called doing a BREAK. Many RUNOFF commands, because of what they do, cause a break in the source file. This action of doing a break without there being a BREAK command is called doing an implicit BREAK.

All the line-insertion commands cause an implicit BREAK before they are processed.

The format of the BREAK command is:

.BREAK

The SKIP command does an implicit BREAK and then skips n printing lines.

The format of the SKIP command is:

.SKIP [n]

If n is omitted, RUNOFF will skip 1 line.



These names of virtue, with  
their precepts, were:  
.skip 2  
.>1. TEMPERANCE.  
.skip 1  
Eat not to dullness; drink not to elevation.  
.skip 2  
.>2. SILENCE.  
.skip 1  
Speak not but what benefit others  
other yourself; avoid trifling conversation.  
.skip 2  
.>3. ORDER.  
.skip 1  
Let all your things have their places; let  
each part of your business have its time.

These names of virtue, with their precepts,  
were:

1. TEMPERANCE.

Eat not to dullness; drink not to elevation.

2. SILENCE.

Speak not but what benefit others other  
yourself; avoid trifling conversation.

3. ORDER.

Let all your things have their places; let  
each part of your business have its time.

The PARAGRAPH command tells RUNOFF to do an implicit BREAK, and then indent m spaces after skipping n printing lines.

The format for the PARAGRAPH command is:

.PARAGRAPH [m] [n]

The default values for m and n are 0 spaces, 1 line. If no values are given, RUNOFF will use the most recently specified values.

"...You were not to be found, and I was  
fearful of losing a bargain."  
.paragraph 5 1  
"Amontillado!"  
.paragraph  
"I have my doubts."  
.paragraph  
"And I must satisfy them."  
.paragraph  
"Amontillado!"  
.paragraph  
"As you are engaged, I am on my way to Luchresi.  
If anyone has a critical turn it is he..."

"...You were not to be found, and I was  
fearful of losing a bargain."

"Amontillado!"

"I have my doubts."

"And I must satisfy them."

"Amontillado!"

"As you are engaged, I am on my way  
to Luchresi. If anyone has a critical  
turn it is he..."

#### Note

In both FILL and ADJUST Modes, if the first character of  
an input line is a space, RUNOFF will act as if you gave  
a PARAGRAPH command here. This is known as implicit para-  
graphing.

Here's an example of BREAK, SKIP, and PARAGRAPH.

```
.paragraph 5 0  
Mr. Bixby made for the shore and soon  
was scraping it, just the same as if it had been  
daylight. And not only that, but singing:  
.break  
.skip 1  
.>"Father in heaven, the day is declining," etc.  
.skip 1  
It seemed to me that I had put my life in the  
keeping of a peculiarly reckless outcast.  
Presently he turned on me and  
said:  
.paragraph  
"What's the name of the first point above New Orleans?"
```

Mr. Bixby made for the shore and soon was  
scraping it, just the same as if it had been  
daylight. And not only that, but singing:

"Father in heaven, the day is declining," etc.

It seemed to me that I had put my life in the  
keeping of a peculiarly reckless outcast.  
Presently he turned on me and said:

"What's the name of the first point above New  
Orleans?"

The EJECT command signals the end of a page, in the same way that BREAK  
signals the end of a line.

The format of the EJECT command is:

```
.EJECT
```

The EJECT command does an implicit BREAK, and skips to the beginning of a  
new page. There are several commands which do an implicit EJECT (which  
includes doing an implicit BREAK).

#### Output Commands: The TTY Command

The most important output command is the TTY command. The TTY command  
causes RUNOFF to display the processed output file on your terminal. If  
you have a printing terminal, you will get a "hard copy" of the output  
file.

The format of the TTY command is:

.TTY

You must specify TTY if you want the output displayed. It can appear anywhere in your source file.

Here's a sample RUNOFF source file which uses all the commands that you have learned:

```
.*This file contains the monthly report of the
.*Universal Mfg. Corp.
.tty
.header/Monthly Report/UNIVERSAL MFG CORP/Page #/
.footer//In-House Only//
.tab @ 10 22 40
.space 2
.skip 1
.>UNIVERSAL MANUFACTURING CORPORATION
.>Report for the month of: July 1984
.space 1
.+Summary
.fill
.paragraph 5 1
Although sales this month have not met up with previous
expectations, our staff in Product Development have
come up with some real rousers which, we are
pleased to announce, should guarantee an upward-spiraling
sales trend for the year to come.
.paragraph
These new products include:
.skip 1
./Euphorics/Domestic Androids/Microwave Freezers/
./Cloned Pets/Reversible Umbrellas/Metric Boots/
./Smart Cars/Wide-beam Lasers/Pet Moon Rocks/
./Portable Windmills/Ultimate Mousetraps/Air Sensors/
./Thinking Caps/Ersatz Sawdust/Presidential Mugs/
./Biorhythmic Sheet Music/"Sadness"/Featherless Chickens/
.skip 1
.paragraph
The schedule of releases is:
.skip 1
.nfill
@MONTH @PHASE @PRODUCTS
.skip 1
@September @Rumours @Androids, freezers, rocks
@@Press Kit @Boots, cars, pets
.skip 1
@October@Follow-up@Freezers, rocks, cars
@@Public demo@Cars, umbrellas
.skip 1
.adjust
.paragraph
```

In addition, Universal Manufacturing is introducing an entire new line of products---complete tailored-to-order universes in customer's choice of Einstein, de Sitter and Lobeshevsky curvature tensors.

.break

.+\*\*\*\*\*WE WILL MAKE 'NESTED' CUSTOM ORDERS\*\*\*\*\*

Our only stipulation will be that we must be part of any and all orders.

.paragraph

In preparation for the necessary changes in our sales approach, we will be issuing new brochures, with covers showing the

five-color map solutions in bright primary inks.

.paragraph

We expect all salespersons to report for re-orientation within two weeks.

The results from processing this sample are shown on the next page in Figure 5-3.

If this file is called REPORT, the commands

```
RUNOFF REPORT
FILE *REPORT
```

will yield the file \*REPORT, as shown on the following page.

Now that you've learned the basic RUNOFF commands, you're ready to process your source file by running RUNOFF.

#### RUNNING RUNOFF (PROCESSING YOUR SOURCE FILE)

Once you've put the desired RUNOFF commands into your source file, you are ready to process the file through RUNOFF.

The sequence for processing a source file is:

1. If you have finished EDITing your source file by inserting RUNOFF commands, but are still in EDITOR, save this file by giving EDITOR the FILE command. (Make sure you give a new filename, unless you are willing to delete the previous file by this name.)

```
file form.letter
OK,
```

2. Give PRIMOS the RUNOFF command. This tells PRIMOS to get the specified file and process it using RUNOFF. The format of the RUNOFF command is:

```
RUNOFF filename
```

Monthly Report

UNIVERSAL MFG CORP

Page 1

## UNIVERSAL MANUFACTURING CORPORATION

Report for the month of: July 1984

## Summary

Although sales this month have not met up with previous expectations, our staff in Product Development have come up with some real rousers which, we are pleased to announce, should guarantee an upward-spiraling sales trend for the year to come.

These new products include:

Euphorics	Domestic Androids	Microwave Freezers
Cloned Pets	Reversible Umbrellas	Metric Boots
Smart Cars	Wide-beam Lasers	Pet Moon Rocks
Portable Windmills	Ultimate Mousetraps	Air Sensors
Thinking Caps	Ersatz Sawdust	Presidential Mugs
Biorhythmic Sheet Music	"Sadness"	Featherless Chickens

The schedule of releases is:

MONTH	PHASE	PRODUCTS
September	Rumours Press Kit	Androids, freezers, rocks Boots, cars, pets
October	Follow-up Public demo	Freezers, rocks, cars Cars, umbrellas

In addition, Universal Manufacturing is introducing an entire new line of products—complete tailored-to-order universes in customer's choice of Einstein, de Sitter and Lobeshevsky curvature tensors.

\*\*\*\*\*WE WILL MAKE 'NESTED' CUSTOM ORDERS\*\*\*\*\*

Our only stipulation will be that we must be part of any and all orders.

In preparation for the necessary changes in our sales approach, we will be issuing new brochures, with covers showing the five-color map solutions in bright primary inks.

We expect all salespersons to report for re-orientation within two weeks.

In-House Only

Figure 5-3. Sample RUNOFF Report

You may give the name of the file that you just finished EDITing, or the name of any other file in your current directory that contains RUNOFF commands. PRIMOS will respond with the word GO, and put you into RUNOFF. RUNOFF will announce itself by displaying the word "RUNOFF" plus the version number of RUNOFF which is on your system, like so:

OK, runoff form.letter

GO

RUNOFF REV 14.0

3. If you did not specify a filename when you gave the RUNOFF command, RUNOFF will immediately ask for a source file, like so:

INPUT FILE:

At this point, give the name of the file that you want processed. If you misspell the filename -- i.e. RUNOFF cannot find any file by that name in your current idrectory -- you will be returned to PRIMOS with the ER! prompt, and will have to give the RUNOFF command all over again.

4. Once RUNOFF finds the specified source file, it enters Command Mode, which it indicates by displaying the word "COMMANDS". You may now give RUNOFF commands directly from your terminal. These commands entered in Command Mode need not be preceeded by periods.

RUNOFF signals it will accept a command by displaying the Command Mode Prompt, which is the Dollar Sign (\$). When you are done giving additional RUNOFF commands, and are ready for your source file to be processed, hit the Carriage-Return Key. This tells RUNOFF to begin processing, which RUNOFF will acknowledge by displaying the word "PROCESSING..." This part of the sequence looks like:

COMMANDS

\$header/UNIVERSAL MFG CORP/NEW PRODUCTS/Page #/

\$spaces 2

\$

PROCESSING...

5. In order to process your source file, RUNOFF needs to know the name of the output file it is about to create. If you have not given an output filename in the source file, RUNOFF

will ask for one:

ENTER OUTPUT FILE TREENAME:

You may specify an existing name, if you want to overwrite a previous version. RUNOFF will ask you to verify that you want to delete the previous version by querying:


OK TO DELETE OLD filename ?

If you answer anything but YES (Y,YE,OK,etc.), RUNOFF will say:

PLEASE SPECIFY NEW FILENAME:

6. RUNOFF will process the source file and create an output file. This output file will be placed into your current directory. If you give the TTY command, it will be displayed at your terminal. You can also inspect it via EDITOR, SLIST, or SPOOL. If there were no command errors in your source file, RUNOFF returns you to PRIMOS with an OK prompt. (RUNOFF Command Errors are explained below.)

Here's a quick summary of the processing sequence:

- 
1. Input, edit and file a source text file which includes RUNOFF commands, using EDITOR.
  2. Give PRIMOS the RUNOFF command.
  3. If you did not give the name of the source file, do so in response to the prompt INPUT FILE:.
  4. Give RUNOFF commands from terminal, if desired. Hit the Carriage-Return Key immediately after the \$ prompt to begin processing.
  5. Give name of output file, if not previously done via RUNOFF's FILE command.
  6. Inspect output file. If there were no command errors, proofread text. If there were, return to the source file and make command corrections, via EDITOR.



The sequence of processing might look like this on your terminal:

```
OK, runoff form.letter
GO
RUNOFF REV 14.0
COMMANDS
$ space 2
$ TTY
$ (CR)
PROCESSING...
ENTER OUTPUT FILE TREENAME: $form.letter

OK,
```

#### RUNOFF COMMAND ERRORS

RUNOFF Command Errors are errors which you make in giving RUNOFF commands. (RUNOFF cannot detect errors in your source text; this will be processed exactly as it appears in your source file. It's up to you to catch errors in spelling, punctuation, grammar, et cetera.)

Command Errors can occur either in your source file or when you are giving commands in RUNOFF's Command Mode. The errors made when in Command Mode are detected instantly; those in the source file are detected as the line which contains the error is processed.

A common source of errors is unintentional characters or words in your source file which you inadvertently entered in EDITOR by trying to give commands while you were in INPUT Mode. If you did not remember to delete these unwanted lines, you will get unexpected errors when you try to process the file through RUNOFF.

There are two kinds of command errors: illegal commands and unrecognized commands.

#### Illegal Commands

Illegal commands are commands which RUNOFF can understand but not carry out. For example, the commands .TAB @ 7 30 12 or .PARAGRAPH 300 can be interpreted, but the values of the parameters are invalid.

RUNOFF calls to your attention any illegal commands you have given by printing out the entire command followed by the message:

```
ILLEGAL COMMAND
```

at your terminal.

### Unrecognized Commands

Unrecognized commands are commands which RUNOFF does not recognize. The typical reasons for this event are that you either misspelled a RUNOFF command word, or used a non-RUNOFF command by mistake - e.g., ATTACH, WHERE, DUNLOAD.

RUNOFF calls to your attention any unrecognized commands by printing out the command word followed by the message:

UNRECOGNIZED

at your terminal.

For both illegal and unrecognized commands, RUNOFF will insert, at the current line of the output file in between a pair of blank lines, the line from your source file which caused the error, like so:

### What To Do When RUNOFF Detects a Command Error

When RUNOFF detects a command error, after printing out the bad command and the error message, it returns to RUNOFF Command Mode. Your choices at this point are:

- Type QUIT, and correct the error in the source file via EDIT.
- Type the correct command, plus a Carriage-Return to continue the processing.
- Type a Carriage-Return to continue processing without correcting the error.

In order to get an error-free output file, you will have to make the appropriate corrections in the source file, by means of EDITOR. The only reason you might have for continuing to process a source file once RUNOFF detects an error is to locate any further errors during the same attempt. Any output file which contains error messages will be useful only for determining the results of your errors.

If RUNOFF detected any errors in your source file, it will return you to PRIMOS by this prompt:

\*\*\*\*\*ERROR(S)\*\*\*\*\*  
ER!

instead of the regular "OK", prompt.

## DONE: THE PROCESSED OUTPUT FILE

Now you're done. You've created a source file of text and RUNOFF commands, and processed this file through RUNOFF to get an output file of formatted text.

All that remains for you to do is proofread your RUNOFF output -- to see if the format suits you, to make sure there aren't any spelling or formatting errors, etc. If you do find things you wish to change, the proper way to do it is:

1. Give the ED command, plus the name of the source file.
2. Make appropriate change in the source file.
3. FILE this, under the old name, or a new one.
4. Give the RUNOFF command, with the name of the corrected source file.
5. Check the new version of the output file.
6. Repeat steps 1-5 until you are satisfied.
7. Make a "hard" copy of the output file, either on the line printer via the SPOOL command, or on a printing terminal by giving the TTY command to RUNOFF or using the SLIST command.
8. Keep whichever of the source and/or output files you want; delete those you do not want.
9. Check your current directory with a LISTF command to make sure you have only those files you want.
10. You can now log out, or proceed to a new task.

## SECTION 6

## MORE RUNOFF

## INTRODUCTION

Before you read through this section, you should be familiar enough with the material in Section 5, THE ESSENTIALS OF RUNOFF, to use those basic RUNOFF commands without any trouble.

This section explains the remaining RUNOFF commands, excluding those which deal with tables of contents and decimalization. The commands are grouped by function, and are explained sufficiently for you to be able to use them. If you need more information about any given command, consult the RUNOFF REFERENCE SECTION.

The remaining commands fall into these categories:

- Page-formatting - specify page size, margins, indentations, text columns, and even & odd headers & footers.
- Output options - control the manner in which you get output.
- Special characters, symbols and conventions - control RUNOFF's reserved characters, definable symbols, and underlining.
- Art and File-insertion - reserve space for art and tables, and insert other source files into the output file.
- Indexing

## PAGE-FORMATTING COMMANDS

RUNOFF's Page-Formatting commands allow you to define any and all of the following:

- Length and width of physical page.
- Top, bottom, side and inter-column margins.
- Left and right indentation.
- Number of text columns per page.
- Different headers and footers for even and odd numbered pages.

Of course, you don't have to give any of these commands if you want to use the default format of the page, (See Figure 5-1).

### Page Size Commands

You can redefine the physical size of the page with these two commands:

`.LENGTH [n]`

`.WIDTH [m]`

The LENGTH command defines the length of the physical page, including the top and bottom margins, as n lines. If no value of n is specified, RUNOFF uses the default value of 66 lines. (At the standard of 6 lines = 1 inch, 66 lines = 11 inches.) The maximum allowable length is 132 lines (22 inches).

The WIDTH command defines the width of the physical page, including the left and right margins, as m spaces. If no value of m is specified, RUNOFF uses the default value of 85 spaces. (RUNOFF works in Pica spaces, which are 10/inch. This is known as "10-pitch type". Although you may be able to vary the type pitch on certain hard copy printers, RUNOFF will always act as if it is printing 10-pitch type.)

At 10 spaces = 1 inch, 85 spaces = 8½ inches. The maximum allowable page width is 170 spaces (17 inches).

Both the WIDTH and LENGTH commands cause an implicit BREAK and EJECT. The page size, therefore, is 8½ by 11 inches.

### Margin Commands

You can redefine any of the margins on the page by means of these commands:

`.TMARGIN [n]`

`.BMARGIN [n]`

`.SMARGIN [m]`

`.CMARGIN [n]`

The TMARGIN command sets the top margin to n lines from the top of the physical page. If n is omitted, the default value of 7 lines is used.

The BMARGIN command sets the bottom margin to n lines from the bottom of the physical page. If n is omitted, the default value of 5 lines is used.

The SMARGIN command sets the side margins (i.e., left and right) to m spaces from each side of the physical page. If m is omitted, the default value of 7 spaces is used.

The CMARGIN command can only be used when you have at least 2 columns of text on the page. (See the COLUMNS command.) This command sets the inter-column margin -- i.e. the number of spaces between each column -- to m spaces. If m is omitted, the default value of 5 spaces is used.

All of the margin commands cause both an implicit BREAK and EJECT. The default margin settings, if none of these commands are given, are:

Top = 7 lines

Bottom = 5 lines

Sides = 7 spaces

Inter-column = 5 spaces (if at least two columns of text are specified)

### Indentation

The indentation commands allow you to move the left and right margins individually and temporarily. Indentation is always relative to the most recent setting. The indentation commands are:

.INDENT [*m*]

.RINDENT [*m*]

.UNDENT [*m*]

.RUNDENT [*m*]

The INDENT command moves the current left margin *m* spaces to the right. If *m* is omitted or zero, RUNOFF will indent by the default value of 5 spaces.

The RINDENT command moves the current right margin *m* spaces to the left. If *m* is omitted or zero, RUNOFF will use the default value of 5 spaces.

The UNDENT command moves the current left margin by *m* spaces to the left. If *m* is zero or omitted, RUNOFF will reset the left margin back to the value specified in the most recent SMARGIN command, or, if none was given, to the default margin of 7 spaces from the side of the page.

The RUNDENT command moves the current right margin by *m* spaces to the right. If *m* is omitted or zero, RUNOFF will reset the right margin back to the value specified in the most recent SMARGIN command, or, if none was given, to the default margin of 7 spaces from the side of the page.

### Columns of Text

The COLUMNS command allows you to have more than one column of text on a page. The format of the COLUMNS command is:

.COLUMNS [*i*]

The default setting is 1 column of text per page. If you use COLUMNS to define more than 1 column of text, the default margin between these columns is 5 spaces. This can be changed via the CMARGIN command.

The COLUMNS command causes both an implicit BREAK and EJECT.

#### Even and Odd Page Headers and Footers

You can specify different headers and footers for the even and odd-numbered pages by using the following commands:

.EFOOTER /left-text/center-text/right-text/

.EHEADER /left-text/center-text/right-text/

.OFOOTER /left-text/center-text/right-text/

.OHEADER /left-text/center-text/right-text/

The rules for these commands are the same as for the header and footer commands:

1. Any character not in the text portions may be used as the delimiters;
2. Text portions may be omitted;
3. All four delimiters must appear in a given command;
4. Each command affects all pages after it is given.

#### OUTPUT OPTIONS

Output Option commands determine what RUNOFF will do with the output file. These choices include:

Specifying a name for the output file, or having no output file.

Process only certain pages of the output file.

Numbering the lines of the output.

Pausing and/or perforating between each page of output.

Suppressing command error messages.

Output option commands are usually given in RUNOFF's command mode.

### Output Filename

The FILE command specifies the name of the output file.

The format of the command is:

.FILE [filename]

If you do not specify a filename in the command, RUNOFF will inquire:

ENTER OUTPUT FILE TREENAME:

If you give a name which already exists in your current directory, RUNOFF will ask:

OK TO DELETE OLD filename?

If you answer anything but YES (Y, YE, O, or OK), RUNOFF will ask for a name for the output file.

### To Get No Output File

The NFILE command tells RUNOFF that you do not want any output file saved.

The format of this command is:

.NFILE

RUNOFF will process the source file, but not save the results. This command has little use, except when you are also giving the TTY command.

### Processing Selected Pages

The FROM and TO commands permit you to select specific range of pages to be processed from a longer source file.

The format of the command is:

.FROM [i]

.TO [i]

You may give either, both or none of these two commands.

If you give the FROM command, page i of the processed output will be the first page displayed and/or filed. If you do not give the FROM command, RUNOFF begins with page 1.

If you give the TO command, page i will be the last page of the processed output to be displayed and/or filed. If you do not give the TO command, RUNOFF will continue until the entire source file has been processed.



If you have reset the page numbering by means of the PAGEN command, RUNOFF will compare the specified number(s) against this page count; otherwise, RUNOFF will compare the number(s) against the sequential page count. (The page number need not physically appear on the pages for RUNOFF to be able to use it.)

### Numbering the Lines of Output

The SOURCE command tells RUNOFF to put line numbers which correspond to lines in the source file by the side of the output file lines.

The format of the SOURCE command is:

```
.SOURCE [n]
```

When the SOURCE command is given, RUNOFF begins generating a list of line numbers one space to the right of the right margin of the output file. Each number corresponds to a non-blank text line from the source file; command lines from the text file, as they do not appear in the output file, are not numbered. If you are making multi-column text, each column will have corresponding line numbers to the right of it.

Each line number will be n greater than the previous one. If the value of n is 1 or omitted, the line numbers will increase by 1. If you specify z value of zero (0) for n, RUNOFF will stop inserting line numbers.

### Pausing and/or Making Perforation Marks Between Pages of Output

RUNOFF has two commands which have an effect between pages of output.

The format of these commands is:

```
.PAUSE
```

```
.PERFORATE [n]
```

The PAUSE command tells RUNOFF to pause at the end of each page of the output file until you type a character at the terminal. This permits you to adjust the paper in a hard-copy terminal, which is particularly useful if you want to run a file off on separate sheets of paper, instead of on the continuous sheets which many terminals use. Typing any character will signal RUNOFF to process another page; we recommend a non-printing character (such as Carriage-Return), which will not leave a mark on the paper.

The PERFORATE command causes RUNOFF to print a line of hyphens, which are meant to act as perforation marks, at the place where the physical bottom of the page should be. This helps to indicate where new pages begin on long rolls of paper output. If you give any value for n that is a positive integer -- e.g., 1, 2, or 378 -- RUNOFF will print one hyphen at each end of the line, instead of a full line of hyphens. The full line is the default.

The PAUSE and PERFORATE modes can be turned off with the commands:

.NPAUSE

.NPERFORATE

respectively.

#### Supressing RUNOFF Command Error Messages

As a rule, you want to know where in your source file you have made RUNOFF command errors, there will be times when you want RUNOFF to process your source file without stopping to display erroneous lines.

The command:

.ERRGO

tells RUNOFF to suppress all error messages as well as the associated actions, and instead to process the entire file and make the output file complete with the results of any errors that occur.

This method of using RUNOFF is primarily useful when you are processing a file as a phantom user -- i.e., you have told the computer to RUNOFF a file during a period of time when you are not going to be at the terminal, and therefore cannot correct and continue processing as errors occur.

The command:

.NERRGO

returns RUNOFF to its default mode of displaying error messages and returning to Command mode.

#### SPECIAL CHARACTER, SYMBOLS AND CONVENTIONS

RUNOFF has a number of special characters with specific functions. There are commands to change the value of these reserved characters.

The commands to change the value of reserved characters are BLANK, ERASE, HYPHEN, KILL and SYCHAR.

The format for the BLANK command is:

.BLANK character

The BLANK command resets the value of the Blank Character. Each Blank character indicates a required blank in the source file and indicates words which must not be broken among lines; these blanks will then be neither suppressed nor padded during FILLing and ADJUSTing.

The formats for the ERASE and KILL commands are:

.ERASE character

.KILL character

RUNOFF's default settings for the ERASE and KILL characters are the same as those used by PRIMOS and EDITOR -- unless, of course, they have been changed on your particular computer. The default value of the Erase Character is the double-quote ("); the default value of the Kill Character is the question-mark (?). These two characters can be used only in RUNOFF's Command Mode. Otherwise, those literal double-quotes and question-marks in your source text would be interpreted as Erase and Kill commands. You can give the value-changing commands BLANK and KILL within your source file.

#### Note

Once you have changed either of these characters, you cannot return them to the default setting within the RUNOFF session. They will be reset when you leave RUNOFF.

The format for the HYPHEN command is:

.HYPHEN character

This command defines the value of RUNOFF's Phantom Hyphen character. The phantom hyphen can be used to indicate places where RUNOFF may hyphenate a given word of text if the word is too long to fit on an output line. The use of phantom hyphens is not recommended, as they can increase the time it takes RUNOFF to process a source file by a significant amount. See the RUNOFF REFERENCE SECTION for a fuller explanation of this character and command.

The SYCHAR command defines the delimiter for Symbol names, which are explained in the following paragraph.

The format for the SYCHAR command is:

.SYCHAR character

The default value is the percent sign (%).

#### Defining and Using Symbols

There are many occasions when you want to leave a "hole" in a document which you intend to fill in later. This "hole" could be a page number which refers to a section you have yet to write, the address and name of a customer, the date, or many other things.

RUNOFF allows you to leave such "holes" in your source file, and gives you a way of labelling each of these so that you can define the contents of the missing material elsewhere in your source file, or even in Command Mode. You can label each place to be filled in by means of Symbols.

Each symbol has a name and a value. You insert the name enclosed in per-cent signs at the appropriate place in your source file, and define the value of a symbol which has a specific name either in Command Mode or within the source file prior to the use of the symbol's name. RUNOFF inserts the specified value of a symbol in the output file whenever its name appears in the source file.

You define the name and value of each symbol by means of the DEFINE command.

The format of this command is:

```
.DEFINE symbol-name value
```

Whenever RUNOFF finds symbol-name enclosed in per-cent signs in your source file, it will replace it, in the output file, with the corresponding value as defined by a DEFINE command. The value may be text, a parameter, or even a command.

For example, the source file statements:

```
.DEFINE Name Arthur Ingrid
.+Dear %Name%,
```

will be processed as:

```
Dear Arthur Ingrid,
```

The rules for defining symbols are as follows:

1. RUNOFF distinguishes between upper and lower case letters in symbol-names. This means that:

TITLE, Title, and title are different symbol names.

2. The name of a symbol may be of any length -- however, RUNOFF will only look at the first six letters. This allows you to use longer names for your own reference, but note, for example, that the names:

Expedite and Expedition would appear to be the same.

3. You cannot use commas, parentheses or spaces in a symbolname.
4. The symbol-value may contain any characters, and may be up to thirty characters in length.

5. You may have up to sixty different symbols defined at a given time. If you need additional symbols, you may redefine an existing symbol-name whose old value is no longer needed, via the DEFINE command. You can undefine one or all symbols without giving new values with the UNDEFINE command. (See the RUNOFF REFERENCE SECTION for details.)

To insert a literal SYCHAR in the text; e.g., 95% pure, type two SYCHARs together: i.e. 95%% pure.

### Special Conventions

RUNOFF has special conventions to cover two particular text-formatting circumstances: beginning a line with a period, and underlining.

Initial Text Periods: Since a period at the beginning of a line signals that the line is a RUNOFF command, you need a special way to indicate that you want a text line to begin with a period. You do this by typing two periods in a row; RUNOFF will process this pair as a single text period which begins a line of text in the output file.

So, for example, the lines:

```
..INSERT FOUR
..PARA
```

would be processed to:

```
.INSERT FOUR
.PARA
```

Underlining: RUNOFF permits you to underline any portion of a line of text. A pair of left braces ({{) indicates the beginning of text to be underlined; a pair of right braces (}}) indicates the end.

For example:

```
{{Underline me}}
```

would be processed to:

```
Underline me
```

RUNOFF allows you to underline:

- Punctuation Marks
- Parts of words
- Centered Text
- Headers, footers, and apportioned text
- Blanks
- Decimal headings

as well as the standard words, sentences, lines and paragraphs.

You cannot underline RUNOFF commands. This means you must be careful not to enclose any commands within the double braces when underlining multiple lines of text.

Headers, footers, and apportioned text can be underlined

by portion:

```
/{{text}}/text/text/
```

or completely -- the entire line:

```
/{{text/text/text}}/
```

For the second case, the outer delimiters must enclose the double braces.

For example:

```
.HEAD/{{Monthly Report/UNIVERSAL MFG CORP/Page #}}/
```

would be processed to:

```
Monthly Report          UNIVERSAL MFG CORP          Page 1
```

---

You can underline blanks ( ) in any of the following ways:

1. In any mode, use the Blank Character within the braces, e.g.

```
{{#####}}
```

2. In NFILL Mode, use regular or reserved blanks, e.g.

```
{{      }} or {{#####}}
```

3. In either mode, use the Underscore Character ( \_ ), e.g.

---

If you want to enter a pair of left or right braces literally into your output file, i.e. {{ or }}, type a Phantom Hyphen between the two braces in your source file.

For example, the source file:

```
You tell RUNOFF to underline text by enclosing it in a
pair of left and right braces, like so:
```

```
.sk 1
.{.RUBOUT.{test}.RUBOUT.}
.sk 1
to get this output:
.sk 1
.>{{text}}
```

is processed to:

```
You tell RUNOFF to underline text by enclosing
it in a pair of left and right braces, like
so:
```

```
{{text}}
```

to get this output:

text

The default value for Phantom Hyphen is the Rubout Key, printed once.

#### Note

1. If your terminal cannot format underlined text properly because it cannot backspace, a separate line containing the underline characters will be printed below the text to be underlined.

3. For Diablo Printers, the codes Control-Left-Parenthesis and Control-Right-Parenthesis will also generate the Left and Right Braces, respectively.
4. If you have underlined a decimal heading, the decimal number will also be underlined, unless an extra space is between the decimal heading command and the text of the heading.

#### BLOCKS OF TEXT, ART AND INSERTED FILES

RUNOFF has commands which allow you to reserve groups of lines on pages for blocks of text or artwork, plus commands which allow you to insert text from other source files.

##### Blocks of Text

There are two RUNOFF commands that reserve blocks of lines on the page, for text which cannot be split among two pages.

The NEED command indicates that the next n lines must appear in a continuous group on the page.

The format of the NEED command is:

.NEED [n]

When RUNOFF processes a NEED command, it immediately checks to see if there are n lines left on the page before the bottom margin. If this proves to be the case, then RUNOFF proceeds as usual. If, however, there are not n lines remaining, RUNOFF immediately does a BREAK and EJECT and then puts the n lines of text at the beginning of a new column or page.

If n is zero or omitted, the default value of 1 is used.

The WIDOW command permits you to specify a minimum number of lines in a paragraph which may appear at the bottom of a page.

The format of the WIDOW command is:

.WIDOW [n]

Where the NEED command is specific, WIDOW is general. Once you give the WIDOW command, RUNOFF checks the text for the bottom n+1 lines of each page of output before finishing the processing of the page. If there is a BREAK, SKIP, or PARAGRAPH command in the corresponding source text, RUNOFF will do an EJECT just before such commands, and continue processing the text on a new page.



## Art

The PICTURE command allows you to save a block of blank lines in a continuous block. This is particularly useful if you plan to insert items into your document which cannot be produced by RUNOFF. This includes artwork, drawings, tables set in different type, and examples.

The format of the PICTURE command is:

.PICTURE [n]

The n in this command always refers to physical lines -- not n times the current value of SPACE. When the PICTURE command is given, RUNOFF will see if n lines remain on the page. If this is the case, RUNOFF will fill the current output line, if you are in FILL mode, fill & adjust the line if you are in ADJUST Mode, and then do a BREAK and SKIP n physical lines.

If n physical lines are not available on the page, RUNOFF continues to process text, and SKIPS n physical lines at the top of the new page before continuing to process text. If n was larger than the number of physical lines between the top and bottom margins, RUNOFF will skip additional lines and/or pages until a total of n lines are skipped.

You can specify more than one picture request at a time. RUNOFF will keep track of up to ten values of n at the same time, and will attempt to put one per page as soon as possible, in the order that each picture was specified; each page will be filled with text below the reserved space.

## Inserting Files

RUNOFF has two special commands which allow you to use additional source files to create a single output file. This is particularly useful for doing tasks such as inserting a name and address from a file list into a form letter, inserting tables and examples which can be done by RUNOFF (or else save space for them with the PICTURE command, and paste them in), and so on. The commands which insert other source files are INSERT and FLOAT. The INSERT command inserts an entire file; the FLOAT command inserts sections of a file.

The format for the INSERT command is:

.INSERT filename

INSERT and FLOAT are explained fully in the RUNOFF REFERENCE SECTION.

## INDEXING

RUNOFF has two commands which permit you to build a file of index notes: the IXFILE command, which defines the name of the file, and the INDEX command, which indicates items in your source file to be noted.

The format of the IXFILE command is:

.IXFILE filename

This defines the name of the index file.

The format of the INDEX command is:

.INDEX string

If you have given the IXFILE command, RUNOFF will compile a list of all strings given in INDEX commands, together with the page numbers of the output page that RUNOFF was processing when each INDEX command was given. Note that this is not a proper index — you will still have to alphabetize this list, and condense multiple entries to a single occurrence of each string plus all its page references.

If you have given at least one INDEX command in your source file, but not specified IXFILE, RUNOFF will ask for the name of the index file with the prompt:

INDEX FILE

You may give a filename, or else hit the Carriage-Return Key. The latter tells RUNOFF that you do not want an index file this time; RUNOFF will ignore all further INDEX commands in the file this time around. (You can specify No Index File via the NIXFILE command.)

See the RUNOFF REFERENCE SECTION for more information on these commands.

## SECTION 7

## RUNOFF DECIMALIZATION

## INTRODUCTION

RUNOFF can generate headings on different numbered levels, and a table of contents. This is a convenient and useful way in which to organize documents, particularly those of a technical nature. Using RUNOFF's decimalization feature, major sections or chapters are automatically assigned single decimal numbers (1,2,...,n). Subsections under the major headings are automatically assigned numbers of the form 1.1, 1.2, 2.1, 3.20, etc. For each successively lower level, another decimalized level is added to the heading number.

Example:

- 1 Title for Section 1
  - 1.1 First Subsection in Section 1  
Text for Section 1.1.
  - 1.2 Second Subsection in Section 1
    - 1.1.1 Yet another Level
  - 1.3 Third subsection under 1
- 2 Title for Section 2

Decimalization simplifies future additions to a document -- there is no need to renumber existing sections and paragraphs by hand.

Attributes of Decimal Headings and Levels

The Level Number: Associated with a block of text to be processed by the decimalization commands in RUNOFF is a level number (L), which is initially set to one. On the appropriate command, this Level number is incremented by one, and formatting takes place as described below. With another decimalization command (.dd), the level number is decremented by one (or more) and formatting reverts to that of the previous level.

Each time a new level is entered or a new block on the current level is started, a header of the following format is generated:

heading number [heading text]

The heading number is of the form 1, 1.1, 1.2, 1.21, etc. where the number level is equal to the current level. The text of the heading (if any) is user-specified. (If there is no heading the level number begins a new paragraph.)

Decimal Level Formatting: Associated with each decimal heading are four variables:

1. The number of lines to skip before the heading,
2. The number of lines to skip after the heading,
3. The number of spaces to indent the heading, and
4. The number of spaces to indent the following text.

All four of these can be at the same level or can be separately specified for each level. Typical formatting might specify: skip two lines before each heading, skip one line after, and indent each successive level five spaces from the previous level.

#### DECIMALIZATION COMMANDS

The RUNOFF decimalization commands are of three types:

1. Format specifications,
2. Block generation, and
3. Level changing.

#### Format Specification Commands

.DSKIP L n1 n2 -- Skip Lines for Heading

For each heading on level L, skip n1 lines before the heading and n2 lines after. If L is zero, the arguments apply to all levels. If L is specified as \*, the arguments apply to the next lower level. If n2 is zero, no lines are skipped, and no break is generated after the heading \* used when the decimal number is just to start a paragraph). If n1 and n2 are not specified, the skip values revert back to the default values of n1=2 and n2=1.

.DINDENT L m1 m2 -- Set Decimal Level Indents

The indent increments for Level L are set to m1 spaces for heading and m2 spaces for the following text. If L is zero, m1 and m2 are increments which are added to the indent amount for each successive level. For example,

```
.di 0 3 0
```

would result in indents of 0, 3, 6, 9, etc.

```
.di 2 7
```

will cause level 2 to be indented seven more spaces than level 1.

If m1 and m2 are omitted, all increments are set back to the default values of 3 for the heading and 0 for the text. (The indent amount for level 1 headings is always zero unless explicitly set to a new value using .di 1 m1 m2.) If L is specified as \*, the arguments apply to the next lower level.

#### Note

The \* specification for the level number is provided so that a series of blocks at some level can be temporarily switched to new formatting without requiring knowledge of the absolute level of the blocks. A typical sequence might be:

```
.ds * 1 0 (Specify new skip amounts for next level)

.dd First heading on Lower Level

.dn Second heading on Lower Level

.du          (Up to previous Level)

.ds          (Reset Lower Level to default skips)
```

#### Block Generation

.DNEXT [text] Generate Next Block on Same Level

A heading is generated for the next block on the current decimal level. Text is the optional text of the heading.

Example:

#### Input

```
.dn Product Description

.dn Current Field History

.dn Future Developments

.dn Glossary
```

#### Result

```
1.2 Product Description

1.3 Current Field History

1.4 Future Developments

1.5 Glossary
```

In this example, the DN command started a new block, forced an indent, and continued the second level paragraph number.

.DNSUPPRESS [text] Generate Unnumbered Block on Same Level

This command has exactly the same effect as .DNEXT, except that no decimal number is generated.

.DDOWN [text] Go Down a Level and Generate Block

The level number is incremented by one and a heading for the first sub-section on the lower level is generated. [text] is the optional text of the heading.

Example:

#### Input

.dn Glossary

.dd Protocol

.dn Synchronous Communications

#### Result

1.5 Glossary

1.5.1 Protocol

1.5.2 Synchronous Communications

In this example, the DD command forced a break, the start of a new block, an indent and a third-level paragraph number.

.DDSUPPRESS [text] Go Down A Level and Suppress Number

This command has the same effect as .DDOWN except no decimal number is generated.

#### Level Changing

.DUP k -- Pop Up One or More Levels

The level number is decremented by k. If k=0, the level number is decremented by one. If k is greater than the current level, the level number is reset to one (the top level). The indent and skip values revert to those of the new level. This command does an implicit .BREAK. Text can be entered without generating a new block at the new level.

.DLEVEL    L -- Set Level Number

The level number command sets the level at which paragraph numbering will continue to L. An implicit .BREAK is done. If L=0, the level is set to one (first-level).

Example:

Input

```
.dl 1
.dn Heading
.dd Lower Heading
.dn Second Lower Heading
.du
.dn Second Heading
```

Result

```
1 Heading
  1.1 Lower Heading
  1.2 Second Lower Heading
2 Second Heading
```

Heading Number Change

.DRESET n m -- Change the Number of the Heading

The argument n specifies the level of the heading, and the m specifies the number of the heading. Therefore, if a file is Chapter 3 of a manual, to begin the paragraph numbering with 3, type:

```
.DR 1 3
```

followed by the command .DN and the heading.

Example:

Input

```
.dr 1 2
.dn Heading
```

Result

```
2 Heading
```

NOTES ON DECIMALIZATION

The current maximum level number is 8, and the largest possible decimal number is 99.99.99.99.99.99.99. (It is strongly recommended, however, that no more than four levels be used.)

If two decimal headers (at the same or different levels are entered with no intervening text, the total skip between the headers will be the sum of the first header and the skip-before the second header.

Appropriate .NEED commands are generated to avoid widows. At least one line of text is guaranteed to follow the heading.

Since the decimalization commands keep track of the current left margin, care must be taken with left indents and undents. In general, things will work out right if, after an .IN n, the margin is reset to its previous value (with an .UN n) before the next block generation or level changing command.

The defining of .UNDENT has been changed so that specifying .U Ø or just .U will cause the left margin to be reset to the current decimal margin, rather than the absolute left margin.

#### TABLE OF CONTENTS

An automatic table of contents can be generated and put into a specified filename with the command:

.TOFC filename

On this command, RUNOFF will make a table of contents entry into the file filename each time it encounters a .DNEXT, .DNS, .DDOWN, or .DDS decimal heading command. The text of the entry consists of the decimal heading number, the text of the heading, periods out to the right margin and the page number on which the heading appears. If a heading is too long to fit on one line, it will be split at a space and continued indented on the following line.

Example:

1	INTRODUCTION.....	1
1.1	Overview.....	1
1.2	Terminology.....	3
1.2.1	Level 3 Heading.....	4
2	GETTING STARTED PROCEDURES.....	7
2.1	System Commands.....	13
3	EXPLANATION OF A HEADING WHICH IS TOO LONG TO FIT ON ONE LINE.....	17



### Specifying a Level Limit

With the command:

```
.TOFC filename level limit
```

a limit can be placed on the number of levels recorded in the table of contents.

For example:

```
.TOFC CONTENTS 3
```

will record all level 1 headings (1), level 2 headings (1.1), and level 3 headings (1.1.1). The table of contents will not include any headings below level 3.

### Closing the Table of Contents File

By using the command:

```
.TOFC
```

with no arguments at all, the user ensures that the table of contents file will be closed. This command is necessary only if the table of contents is going to be appended to the main file with an `.INSERT` command.

### Turning Off Contents Generation

To omit certain decimal headings from the table of contents, use the two commands:

```
.TOFC Ø
```

```
.TOFC 1
```

When RUNOFF encounters `.TOFC Ø` (off), it will ignore all following decimal headings until it encounters `.TOFC 1` (on).

## DECIMALIZATION COMMAND SUMMARY

<u>Command</u>	<u>Function</u>
. <u>DDOWN</u> [text]	Go down one level and generate a block.
. <u>DDSUPPRESS</u> [text]	Go down one level but suppress the number.
. <u>DINDENT</u> L m1 m2	Set indent increments for level <u>L</u> to <u>m1</u> and <u>m2</u> spaces.
. <u>DLEVEL</u> L	Define the level at which numbering continues as level L.
. <u>DNEXT</u> text	Generate a heading for the next block on the current decimal level.
. <u>DNSUPPRESS</u> text	Generate a heading on the current decimal level, but do not generate a number.
. <u>DRESET</u> n m	Reset the level number to <u>n</u> and the next paragraph number to <u>m</u> .
. <u>DSKIP</u> L n1 n2	For each heading on level L, skip n1 lines before the heading and <u>n2</u> lines after it.
. <u>DUP</u> k	Decrement the level number by <u>k</u> .

## TABLE OF CONTENTS COMMAND SUMMARY

<u>Command</u>	<u>Function</u>
.TOFC filename [level-limit]	Generate a table of contents to the specified filename. If the level limit is specified, only decimal headings down to that level will be recorded in the contents.
.TOFC	Close the current table of contents.
.TOFC 0	Temporarily turn off the generation of the table of contents and at .TOFC 1,
.TOFC 1	turn it on again.

PART 2

REFERENCE

PDR-3104

## SECTION 8

## THE EDITOR REFERENCE SECTION

This section contains complete information on all the EDITOR commands. The commands are listed in alphabetic order.

## EXPLANATION OF THE COMMAND FORMAT

EDITOR's command format is:

COMMAND parameter

The word in capital letters is the command word.

Command Words

The underlined letter(s) of the command words indicates the required abbreviation. You must type at least these letters; you may type as many more as you wish. The following are all acceptable ways to input a command whose format is APPEND:

A  
APP  
APPE  
APPEND

Also, you may type the letters of command words in any combination of upper and lower case letters. All of the following are equally valid:

A  
a  
APpend  
append  
ApPeND  
appEND

Parameters

As a rule, the parameter in a command format will be:

- the word filename, representing a filename
- the letter n, representing a number.
- the word character, representing a single character
- any of the words string, text, or newline, representing a piece of text.

to type a space between the command word and the number. For example, the following are all valid:

```
Print5
Pr5
PRINT 15
p-5
p -5
```

Note that there cannot be a space between the minus sign and a number.

If the parameter is a filename or a character, you must have at least one space between the command word and the parameter, like so:

```
load memo
kill &
```

If the parameter is a text string (indicated by text, string, newline), EDITOR assumes that there is exactly one space between the command word (or abbreviation) and the text string, like so:

```
find DEPARTMENT
append and so forth
insert Dear Sir or Madame,
```

Any space after the first space is considered part of string. So, for example, the commands:

```
append et cetera
find henceforth
```

would be using the string " henceforth" which began with three blank spaces.

APPEND string

The APPEND command attaches the specified text to the end of the current line, like so:

```
print
'Twas brillig and
append the slithey toves
'Twas brillig andthe slithey toves
```

One blank separates the command word APPEND (or whatever abbreviation) from the string you wish to append. All further blanks will be treated as text.

If you want to have a space between the last word of the current line and the first word you are appending, you must type two spaces between the command word and the first word of appended text:

```
print
'Twas brillig and
append  the slithey toves
'Twas brillig and the slithey toves
```

If there is no blank between APPEND and string, EDITOR gives the error message:

```
BAD APPEND
```

and you should try again.

The string to be appended is terminated by either a carriage-return or a semicolon (;).

Note

No trailing blanks -- i.e., blanks between the last non-blank character of appended text and the CR or ; -- are inserted!

You can use commas (,) in the string.

```
print
'Twas brillig and
append , dig it, those slithey toves
'Twas brillig and, dig it, those slithey toves
```

You can NOT append semicolons -- unless the semicolon character has been freed for use as a text character via the SYMBOL command. They must be inserted by using the CHANGE, MODIFY, or GMODIFY command.

```
print
'Twas brillig
append & the slithey toves
'Twas brillig& the slithey toves
change/;/;
'Twas brillig; the slithey toves
```

BOTTOM

The BOTTOM command positions the pointer at the bottom of the EDITOR work file. A PRINT command at this point would show the null line .NULL. any new lines inserted now would go under the last line of text in the file. Any attempt to find the NEXT line would result in the word:

BOTTOM

For example:

```
bottom  
print  
.NULL.  
next  
BOTTOM
```

BRIEF

The BRIEF command suppresses the verification output produced by the following commands:

```
APPEND  
CHANGE  
FIND  
GMODIFY  
LOCATE  
MODIFY  
NEXT  
NFIND  
OVERLAY  
POINT
```

BRIEF Mode allows experienced users to work faster. This is particularly useful on terminals which print at a slow speed.

When in BRIEF Mode, use the PRINT command to check the results of your most recent command, like so:

```
brief  
locate mimsey  
print  
All mimsey were the borogoves  
next  
c/mome/gnome  
print
```

To re-activate the verification responses, use the command VERIFY (which is the default).



CHANGE/string-1/string-2/ [G] [n]

The CHANGE command replaces string-1 with string-2.

```
print
And the gnome raths outgrabe.
change/gnome/mome
And the mome raths outgrabe.
```

The first character after the command word CHANGE (or abbreviation) will be used as the delimiter in this instance of the command. Any character, including the semicolon (;) and the space ( ) may be used as a delimiter instead of the slash. This means that the following sample commands would all have the same effect:

```
change&gnome&mome&
change ;gnome;mome;
change KgnomeKmome
ch  gnome mome
change >gn>m>
```

If the letter G (for General) is specified, CHANGE will change every occurrence of string-1 on a line. If you don't specify G, only the first incidence of string-1 will be changed.

If the value of n is 1, EDITOR will only make changes on the current line. (If n is either 0 or unspecified, the default value of 1 is used.) If a value other than 0 or 1 is specified, EDITOR will inspect and make changes on n lines starting at the current line, and leave the pointer positioned at the nth line. If there are less than n lines in the file below the current line, the pointer will be left at the null line below the last line, and the message BOTTOM will be printed.

EDITOR will print out all changed lines, plus last line examined.

#### Notes

1. Remember to TOP before making changes on the file as a whole.
2. You can omit the closing delimiter (/) if you end the command with a Carriage-Return.
3. You can specify the semicolon (;) as a text character within the delimiters--i.e., if you used "@" every place in your file where you wanted to use ";", then the command sequence

```
TOP, CHANGE /@;/ G 999
```

would change the @'s to ;'s. (Make sure n is greater than the number of lines in your file.)

DELETE [n]

The DELETE command deletes n lines, including the current line, from the EDITOR work file, and leaves the pointer at the place where the last deleted line was. (The line .NULL. will be maintained, in case you wish to insert a new line; this null line will disappear as soon as a new command moves the pointer away.)

```

print
'Twas brillig and the slithy toves
delete
print
.NULL.

```

If n is omitted, the default value of 1 is used, and only the current line is deleted. The value of n may be positive or negative, indicating deletion of the current line plus lines below or above the current line.

Note

Since n always includes the current line,  $D1 = D = D - 1$ .

Helpful Hints: To avoid wiping out file lines completely, use the DUNLOAD command, and delete the DUNLOAD-created file(s) later. If the deleted lines were in the original file, you can QUIT without FILEing and start EDITing a new copy, if you have to. Of course, you will have to do all changes from the previous session on this new copy again.

DELETE TO string

The command DELETE TO string deletes from the EDITOR work file all lines, starting with the current line, up to but not including the first following line which contains string.

```

top
delete to mimsey
All mimsey were the borogoves
top, print 4
.NULL.
All mimsey were the borogoves
And the mome raths outgrabe.
BOTTOM

```

WARNING

Be sure string is there and spelled exactly as specified or you will delete everything in your file below the current line.

There must be a blank between TO and string.

See the Helpful Hints in DELETE.

DUNLOAD filename [n]

The DUNLOAD command creates a new file with the indicated name, copies n lines from the EDITOR work file, beginning with the current line, into this new file, and then deletes these n lines from the work file.

```
print
'Twas brillig and the slithy toves
dunload
print
.NULL.
```

Note

Be careful not to specify a filename currently in use unless you want the old file wiped out. If filename is not specified, you will get the error message:

BAD DUNLOA

If n is not specified, the default value of 1 is used and one line is DUNLOADED.

DUNLOAD leaves the pointer positioned at a null line where the deleted lines used to be; this null line disappears as soon as the pointer is moved.

The DUNLOAD command is useful for moving lines of text to different places; DUNLOAD can also be used instead of DELETE if you want to make sure you don't accidentally delete large chunks of text. (But don't forget to delete the DUNLOAD-created files from your UFD when you're through with them.)

DUNLOAD filename TO string

The DUNLOAD TO command copies lines from the EDITOR work file into a new file named filename and then deletes these lines from the original file. Lines are copied starting with the current line and continuing down until a line containing string is found, or until BOTTOM is reached. There must be a blank between TO and string. If filename is not specified, you will get a file named TO.

```

print
'Twas brillig and the slithey toves
dunload topline 2
file

OK, slist topline
GO
'Twas brillig and the slithey toves
Did gyre and gimbol in the wabe.

```

DUNLOAD TO leaves the pointer positioned at a null line where the deleted lines used to be; this null line disappears as soon as the pointer is moved.

The DUNLOAD command is useful for moving lines of text to different places; DUNLOAD TO can also be used instead of DELETE TO if you want to make sure you don't accidentally delete large chunks of text. (But don't forget to delete the DUNLOAD-created files from your UFD when you're through with them.)

Remember:

1. Be very sure the string is there and spelled exactly as specified or you will delete everything in your file below the current line.
2. Don't specify a filename currently in use unless you want the old file wiped out.

ERASE character

The ERASE command allows you to change the previous value of the Erase Character to the character specified; this change will be in effect either until you give a new ERASE command within the EDITOR session or until you QUIT from EDITOR.

The Erase Character rubs out the immediately preceding character of terminal input.

```

erase &
print
'Twas brillig,
append and the slim&they toves
'Twas brillig, and the slithey toves

```

QUITting automatically resets the Erase Character to the default. Unless your System Administrator has changed the System Defaults, the EDITOR Default Erase Character will be the double-quote (") character.

The ERASE command is useful if you intend to use the default character often as a text character. The Erase character of the moment can always be entered as text by preceding it with an up-arrow (↑).

To check the current value of the Erase Character use the PSYMBOL command.

#### FILE [filename]

The FILE command turns the EDITOR work file, (which is so far only a temporary) into a permanent file in your UFD (or sub-UFD), and implicitly returns you from EDITOR back to PRIMOS.

#### WARNING

Since the work file has no existence outside of EDITOR, you must FILE if you want to save work.

The rules for using the FILE command are:

1. If you have been creating a new file, you must specify filename. (The error message FILENAME MUST BE SPECIFIED will occur if you don't.)

#### WARNING

You cannot have two files with the same name in the same UFD!! If you give a filename which already exists in your UFD, EDITOR will delete the old file by that name from your current UFD and put the work file in its place.

```
file
FILE NAME MUST BE SPECIFIED
?
file memol
```

2. The same warning holds true for old files. If you have been working on an old file and you specify the old filename, or say FILE without a filename, your old copy will be deleted, and only your new version kept. Giving a new filename will keep both the old and new versions--but be sure not to accidentally wipe out some other old file by using its name.
3. If you do not wish to save your work from a given session--i.e., want to save your old version, if any--but don't want what you just did in the work file copy--type QUIT instead of FILE. If there is anything in your current file, EDITOR will inquire: FILE MODIFIED, OK TO QUIT? to double-check with you. A YES response QUITs you back to PRIMOS; NO provokes PLEASE FILE, at which point you give the command.

```
quit
FILE MODIFIED, OK TO QUIT? nq
PLEASE FILE
?
file new.memo
```

#### 4. Rules for making filenames

- 1) Filenames can be up to thirty-two characters long.
- 2) The first characters may be any characters except a digit. However, starting with a letter is a good idea.
- 3) Filenames can contain only the following characters:

```
A through Z
0 through 9
- & $ * _ . /
```

Characters NOT permitted in filenames include:

```
Imbedded blanks
Special characters such as: , ? ! @ ;
```

- 4) Upper and lower case letters are treated as upper case by PRIMOS. (Letters inputted in lower-case will be converted to upper-case.)

```
NEWFILE
Todays-Prices
Highs&Lows
$Monthly.Report
R34587
A-Tale-Of-Two-Cities
```

#### Note

The FILE command can be used to make copies of any file simply by typing ED plus filename, and FILEing the copied work file immediately, using a different filename, like so:

```
OK, ed memol
GO
EDIT
file spare-memol
```

OK,

#### FIND string

The FIND command finds the first line below the current line which begins with string, and makes that line the current line.

If no line beginning with string can be found, the pointer stops at the end of the file, and the word BOTTOM is printed.

```
find All  
All mimsey were the borogoves  
print  
All mimsey were the borogoves
```

#### Note

The FIND command distinguishes between upper and lower case letters in string. If you are unable to FIND old lines in your file, but can FIND newly inserted ones and your current display is entirely in upper case letters, the CASE control on your terminal may be in the wrong position.

FIND(n) string

You can also FIND a string starting on other column 1 of the line, by specifying the number of the column within parentheses directly after the command word.

```
find(5) mimsey  
All mimsey were the borogoves
```

#### GMODIFY

The GMODIFY (General Modify) instruction allows characters to be modified on the current line by a series of subcommands specifying the modifications. If the current line is null, GMODIFY will return with the message .NULL.

GMODIFY can be separated from other commands on the same line by a comma or semicolon, and individual subcommands can be separated by spaces. As with all other commands, if GMODIFY detects an error, the remainder of the current command line is ignored and the current line is left intact.

Subcommands consist of a single letter. Some are followed by single characters and others are followed by strings enclosed in delimiters. In the table below c stands for any character, and n for a number greater than zero. The / is used to indicate a delimiter which can be any character not contained in the string, except an Erase, Kill or CR.

<u>COMMAND</u>	<u>FORMAT</u>	<u>MEANING</u>
A	A/string/	Append - copy to .NULL. and append string to end of line.
B	Bn	Back - move INLIN pointer back <u>n</u> spaces but not past beginning of line.
C	Cc	Copy - copy up to but not including <u>c</u> .
D	Dc	Delete - delete up to but not including <u>c</u> .
E	En	Erase - delete next <u>n</u> characters.
F	F	Finish - copy to .NULL.
I	I/string/	Insert - insert string at the current position.
M	Mn	Move - copy <u>n</u> characters.
N	N	Negate - reverses the sense of <u>c</u> in the next C or D command. For example: <u>N</u> Cc will copy up to but not including the first character which is not <u>c</u> . The <u>N</u> can be separated from the C or D by any number of other commands, but will remain in effect only for the first following C or D.
O	O/string/	Overlay - overlays string on the line from the current position. A space leaves the original character unaltered and wild card symbol (!) changes it to a space. All other characters are copied from string. If the overlay extends past the .NULL. all INLIN characters are assumed to be spaces.
R	R/string/	Retype - retypes line with string from the current position. Retype is line an O which does not recognize space and ! as special characters.
S	S	Start - move INLIN pointer to beginning of line.

End-of-line - Although the new line length is limited only by the buffer size, no command will transfer any characters past the end of INLIN.  
For example:

FM9 = F



As a further example:

Initial line: abcXXXX

Command: GMODIFY DXFSM3

Yields XXXabc

#### Notes

1. These GMODIFY commands would accomplish the same result:

```
GMODIFY E3FBACX
GMODIFY DXa/9bc/
GMODIFY DFXSCX
```

2. Mn, Fn, and Bn accept multi-digit n's. For example, G M72 will move 72 characters.
3. I/string/, R/string/, O/string/ and A/string/ do not require the closing delimiter if a CR follows the string on the command line.

INPUT  $\left\{ \begin{array}{l} \text{(ASR)} \\ \text{(PTR)} \\ \text{(TTY)} \end{array} \right\}$

The INPUT command reads text from the specified input devices:

```
ASR = Teletype paper tape reader
PTR = High-speed paper tape reader
TTY - Terminal
```

TTY is the default.

#### INSERT newline

The INSERT commands inserts newline following the current line; the inserted line then becomes the current line.

The first space after the command word separates it from newline.

You may not use semicolons in newline. A semicolon or CR will signal the end of the line being inserted. Unless SYMBOL has been used to make semicolon (;) a valid text character - in which the character that has replaced ; cannot be inserted.

```
'Twas brillig, and the slithey toves
insert Were doing their holiday shopping.
next-l
'Twas brillig, and the slithey toves
print 2
'Twas brillig, and the slithey toves
Were doing their holiday shopping.
Insert On Macy's! On Gimbel's! On Woolworth's and Kresge's!
print
On Macy's! On Gimbel's! On Woolworth's and Kresge's!
```

### KILL character

The KILL command changes the Kill Character from whatever it previously was to the specified character. The Kill Character, when typed, deletes (kills) all the preceding terminal input up to the most recent CR - i.e., allows you to wipe an entire line of typing so that EDITOR never sees it.

The use of the KILL command is to free up the current Kill Character for use in text without putting an up-arrow (↑) before it all the time. You can change the Kill character back with a new KILL command; also, leaving EDITOR automatically resets the default Kill Character (which is the question mark (?), unless your System Administrator has changed it).

There must be exactly one space between the command word KILL and the new Kill Character.

You can check the current value of the Kill Character with the SYMBOL command.

Remember: To insert the current Kill Character as text, precede it with an up-arrow.

```
kill &
insr&insert He tolkh&insert He took his vorpal sword in hand
print
He took his vorpal sword in hand
```

### LINESZ n

The LINESZ command allows the line size of the editor to be changed. In no case can it be made larger than the maximum size (initial size at start-up), but within that limit, it can be set or reset as much as desired. The command format is LINESZn, where n (0<n<n LSMAX) is the characters in the line. If EDLIN, STRA, STRB, STRC, or INLIN contain more than the new maximum characters, they will be shortened appropriately. If the file contains a line longer than the new size, the message BAD LINE IN FILE will be printed and the line broken (one or two characters may be

lost). When a line is typed that is too long, the ? is printed and the BELL is rung, and the line is truncated. In INPUT mode, the remainder of the typed line becomes a new line. The character which caused the overflow, however, is lost.

This command is useful only when you are creating a new file, and should be used before you enter any text. The default value of n is 144.

#### LOAD filename

The LOAD command copies the contents of the specified file into the EDITOR work file just below the current line. The pointer will then be just below the end of the LOADED text, positioned at a null line.

LOAD does not affect the contents of the original file filename in any way; it simply copies the contents of filename into the work file.

Remember: LOADED text will not go in your permanent files in your UFD unless you FILE at the end of the EDITING session.

slit sub.poem

GO

Twinkle, twinkle, little bat,  
How I wonder where you're at.

OK, ed poem

GO

EDIT

print 3

.NULL.

'Twas brillig, and the slithey toves  
Did gyre and gimbol in the wabe.

load sub.poem

EDIT

top, print 5

.NULL.

'Twas brillig, and the slithey toves  
Did gyre and gimbol in the wabe.

Twinkle, twinkle, little bat,  
How I wonder where you're at.

LOCATE string

The LOCATE commands finds the first line below the current line which contains string and makes that line the current line. If no line containing string is found below the current line, BOTTOM will be printed and the pointer left at the end of the file.

```
locate mimsey
All mimsey were the borogoves
```

The first space after LOCATE separates string from the command word. All other blanks will be considered part of string. You may use the Wild Card (!) and Match-n-spaces (#) symbols in string

MODE  $\left\{ \begin{array}{l} \underline{\text{COUNT}} \\ \underline{\text{NCOUNT}} \end{array} \right. \text{ start increment width } \left\{ \begin{array}{l} \underline{\text{PRINT}} \\ \underline{\text{BLANK}} \\ \underline{\text{SUPPRESS}} \end{array} \right\}$

MODE COUNT allows you to increment a counter symbol with every use and replaces it in the text by the current value, whenever the counter symbol occurs in:

The commands: APPEND  
INSERT  
OVERLAY  
RETYPE  
GMODIFY with A, I, O, R subcommands

Input Mode.

The meanings of the COUNT parameters are:

start	initial value for the counter >0 (default = 1)
increment	initial increment $\neq$ 0 (default = 1)
field	field width (number of digits) $1 \leq \text{field} \leq 5$ (default = 5)
PRINT	Print lead zeroes (default)
SUPPRESS	Do not print lead zeroes
BLANK	Replac lead zeroes with blanks

The Counter Character may be redefined via the SYMBOL command. Its default is the @.

Again: the default values for COUNT are:

```
Initial value = 1
Increment      = 1
Field width    = 5
PRINT lead zeroes
```

MODE NCOUNT frees up the count character. The COUNTER character is treated as a normal printing character. This command does not affect the values of the counter, increment, field or the PRINT, SUPPRESS or BLANK parameters.

MODE NCOUNT is default.

MODE { NUMBER  
          NNUMBER }

The command MODE NUMBER causes EDITOR to print the line number (as printed by WHERE) in front of each line of the file when typed out either by verification responses or by the PRINT command.

```
mode number
print 5
.NULL.
00001: 'Twas brillig, and the slithy toves
00002: Did gyre and gimbol in the wabe.
00003: All mimsey were the borogoves
00004: And the mome raths outgrabe.
```

#### Note

The line number is NOT part of the actual file - it is a number EDITOR uses for its own reference. You cannot FIND or LOCATE line numbers; however, you can determine a line's number via WHERE, and go directly to a line via POINT.

The command MODE NNUMBER turns off the display of line numbers. MODE NNUMBER is the default mode.

MODE { COLUMN  
          NCOLUMN }

The command MODE COLUMN causes a column header display to be printed every time you enter Input Mode during an EDITOR session:

```
      1      2      3
123456789012345678901234567890123456 -----> to column 79.
```

mode column

INPUT

	1	2	3	4	5	6	7
1234567890	1234567890	1234567890	1234567890	1234567890	1234567890	1234567890	123456789

The command MODE NCOLUMN turns off the column header display. NCOLUMN is the default mode.

MODE { PROMPT  
NPROMPT }

The command MODE PROMPT causes EDITOR to start printing prompt characters. NPROMPT is the default mode, and is reset every time you leave EDITOR.

mode prompt\$ print3

.NULL.

'Twas brillig, and the slithy toves  
Did gyre and gimbol in the wabe.

\$

INPUT

& But there is no joy in Mudville

&amp;

EDIT

\$ top,print 4

.NULL.

'Twas brillig, and the slithy toves  
Did gyre and gimbol in the wabe.

But there is no joy in Mudville

The default prompts are the Ampersand (&) for INPUT MODE and the Dollar Sign (\$) for EDIT MODE.

MODE { PRALL  
PRUPPER  
PRLOWER }

The Case Modes allow you to indicate upper and lower case letters when inputting and outputting text on terminals which have only upper or lower case display.

Upper-case is signalled by an Up-arrow-U (↑U) before a series of letters which are meant to be upper-case; lower-case by Up-arrow-L (↑L) before lower case.

MODIFY/string-1/string-2/[G] [n]

The MODIFY command changes string-1 to string-2 like CHANGE, but does not affect the spacing of characters in the original line. MODIFY operates in the following manner:

1. Locates string-1 in current line.
2. Starting at the beginning of string-1, replaces with blanks as many characters as there are in string-1 or string-2, whichever is longer.
3. Copies string-2 onto the current line beginning at the first newly-made blank.

```

print
'Twas brillig, and the slithy toves
modify/brillig/July/
'Twas July      , and the slithy toves
modify/July/All-Hallow's Eve/
'Twas All-Hallow's Eve slithy toves

```

The first character after the command word MODIFY (or abbreviation) will be used as the delimiter in this instance of the command. Any character including the semicolon (;) and the space ( ) may be used as a delimiter instead of the slash.

If the letter G (for General) is specified, MODIFY will alter every occurrence of string-1 on a line. If you don't specify G, only the first incidence of string-1 will be modified. If the value of n is 1, EDITOR will only MODIFY on the current line. (If n is either 0 or unsepecified, it is assumed to be 1.) If a value other than 0 or 1 is specified, EDITOR will inspect and MODIFY n lines starting at the current line, and leave the pointer positioned at the nth line.

Note

If there are less than n lines in the file below the current line, the pointer will be left below the last line (at .NULL.) and the message BOTTOM will be printed.

EDITOR will print out all modified lines, plus the last line examined.

Notes

1. Remember to TOP before MODIFYing on the file as a whole.
2. You can omit the closing delimiter if you end the command with a Carriage-Return.
3. You can specify the semi-colon (;) as a text character within the delimiters -- i.e., if you used "@" every place in your file where you wanted to use ";", then the command sequence

TOP, MODIFY/@/;/G9999

would change all the @'s to ;'s. (Make sure n is greater than the number of lines in your file.)



MOVE      buffer-1    { buffer-2 }  
                              /string/ }

The MOVE command moves one line of text into buffer-1 from buffer-2 or string.

The possible buffers are:

EDLIN  
 INLIN (current line)  
 STR1 (or STRA)  
 STR2 (or STRB)  
 STR3 (or STRC)  
 STR4 (no alternate names)  
 .  
 .  
 .  
 STRn

The delimiters around string can be any non-alphabetic character including comma (,) or semicolon (;), which do not appera in string. If MOVE is the last command on the line, the closing delimiter may be omitted.

#### Note

MOVE buffer-1// clears buffer-1.

NEXT [n]

The NEXT command moves the pointer n lines. Positive values of n move the pointer to the bottom, negative values to the top.

If n is 0 or unspecified, the default value of 1 is used.

top, next  
 'Twas brillig, and the slithey toves  
next 2  
 All mimsey were the borogoves  
next 4  
 The jaws that bite, the claws that snatch--  
next -5  
 Did gyre and gimbol in the wabe.  
next 0  
 All mimsey were the borogoves  
next  
 And the mome raths outgrabe.

If a NEXT command would move you to or beyond TOP or BOTTOM, the pointer stops at the appropriate null line.

NFIND string

The NFIND command finds the first line below the current line which does not begin with string, and makes that line the new current line.

```

print 4
'Twas brillig, and the slithey toves
Did gyre and gimbol in the wabe.
All mimsey were the borogoves
And the mome raths outgrabe.
top,next
'Twas brillig, and the slithey toves
nfind Did
All mimsey were the borogoves

```

There must be EXACTLY one space between NFIND and string. If NFIND can't find a line which doesn't begin with string, the pointer will be left at BOTTOM.

Like FIND, you can NFIND beginning on a column other than column 1, like so:

```

NFIND(n) string
top,print 5
.NULL.
'Twas brillig, and the slithey toves
Did gyre and gimbol in the wabe.
All mimsey were the borogoves
And the mome raths outgrabe.
next -3
'Twas brillig, and the slithey toves
nfind(5) gyre
All mimsey were the borogoves

```

Remember: There must be no spaces between NFIND and (n).

Exactly one space between (n) and string. The parentheses ( ) are required.

OUTPUT (DISPLAY)  
(TTY)

While all PRINT commands will print on the user terminal (TTY), the command OUTPUT DISPLAY permits you to send Verification Output to a DISPLAY terminal (which is ...)

OUTPUT TTY is the default; if no device is specified, TTY is assumed.

(Using the DISPLAY command option on this calls for a 9600 baud display terminal to be connected to Port 3 of the System Option Controller. If you don't have one, this command is of no use to you.)

OVERLAY string

The OVERLAY command superimposes the indicated string over the current line, beginning with column 1 as follows:

1. Any character in string EXCEPT !, /, ", ?, or ; will replace the character in the corresponding column of the current line. (Note: ↑" and ↑? are each considered to be a single text character.)
2. An exclamation (!) in string forces a space in the corresponding column.
3. A space ( ) in string leaves the original character unchanged.
4. The tab (\) causes a tabbing to the next tab stop.
5. ", ?, and ; have their usual control functions of Erase, Kill, and End-of-Command string.

```

Did gyre and gimbol in the wabe.
overlay XYZZY
XYZZYyre and gimbol in the wabe.
overlay XX!!!!XX
XX      XXand gimbol in the wabe.
overlay T e c t
TXe c tXXand gimbol in the wabe.
overlay W\Juniper bush!!
WXe cJuniper bush  in the wabe.
overlay Did!macy!and!gimble
Did macy and gimble in the wabe.
overlay      gyre      ol
Did gyre and gimbol in the wabe.

```

PAUSE

The PAUSE command allows you to return to PRIMOS level without ending your EDITOR session. PAUSE preserves EDITOR's work file with all your input and/or editing, and holds the pointer at the current line.

PAUSE is useful, among other things, if you wish to check your UFD for potentially duplicate filenames before doing a FILE. To return to EDITOR, type START -- and a carriage return, of course. You are now back in EDITOR, wherever you left off.

```
OK, ed poem
GO
EDIT
next 2
Did gyre and gimbol in the wabe.
pause

OK, cname mem01 memo3
OK, start
GO
print
Did gyre and gimbol in the wabe.
```

#### Note

You cannot return to a specific EDITOR session via START if:

1. You type ED again
2. You do anything other than LISTF  
CREATE  
DELETE  
CNAME  
ATTACH
3. You log out.

Any of these three actions will have wiped out your EDITOR file.

#### POINT n

The POINT command positions the pointer at line n (makes line n the current line).

The value of n must be >0. If n is greater than the number of lines in the file, the pointer will be left at the bottom.

```

point 3
All mimsey were the borogoves
point 7
The jaws that bite, the claws that snatch--
point2
Did gyre and gimbol in the wabe.
where
LINE          2
point 6
"Beware the Jabberwock, my son!
where
LINE          6

```

### Note

The line numbers are not actually part of your work file; EDITOR generates them for its own reference. You can determine line numbers specifically via WHERE or generally via MODE NUMBER.

### PRINT [n]

The PRINT command prints n lines, including the current line, and makes the last line PRINTed the new current line.

If n is negative, EDITOR first backs up n lines, beginning the count with the current line, and then prints one line.

The space between PRINT and n is optional. A PRINT immediately after the following commands yields .NULL.:

```

BOTTOM
DELETE
DUNLOAD
LOAD
TOP

```

### Note

PRINT = PRINT 1 = PRINT -1 = PRINT Ø

```

print 3
.NULL.
'Twas brillig, and the slithey toves
Did gyre and gimbol in the wabe.
print 6
Did gyre and gimbol in the wabe.
All mimsey were the borogoves
And the mome raths outgrabe.

"Beware the Jabberwock, my son!
The jaws that bite, the claws that snatch--
print
The jaws that bite, the claws that snatch--
print -5
All mimsey were the borogoves
print -1
All mimsey were the borogoves
print -2
Did gyre and gimbol in the wabe.
print 0
Did gyre and gimbol in the wabe.
print
Did gyre and gimbol in the wabe.

```

#### PSYMBOL

The PSYMBOL (Print Symbol) command prints a full list of EDITOR's Special Symbol Functions and their current character values. The list of Special Symbol Functions and System Default character values is:

KILL	?	Erase entire line
ERASE	"	Erase previous character
BLANKS	#	In FIND or LOCATE, match any number of spaces
WILD	!	In <u>string</u> of FIND or LOCATE match any single character. In OVERLAY, turn matching characters into a blank.
TAB	\	Tab
ESCAPE	↑	Up arrow
CPROMPT	\$	Edit Mode Prompt
DPROMPT	&	Input Mode Prompt
COUNTER	@	Counter
SEMICOLON	;	End of Command or input line.

You can change a symbol's value via the SYMBOL command; the ERASE and KILL characters can also be changed by their respective commands.

<u>psymbol</u>	
KILL	?
ERASE	"
BLANKS	#
WILD	!
TAB	\
ESCAPE	^
CPROMP	\$
DPROMP	&
COUNTE	@
SEMICO	;

Remember: EDITOR resets all Special Symbols back to their default values every time you QUIT the EDITOR and/or LOGOUT.

PUNCH     $\left\{ \begin{array}{l} \text{(ASR)} \\ \text{(PTP)} \end{array} \right\} n$

The PUNCH command punches n lines on high-speed (PTP) or low-speed (ASR) paper-tape punch in a form suitable for use by the ED input command. PTP is default. The value of n must be greater than zero.

PTABSET   ptab-1 ptab-2... ptab-8

The PTABSET command tells EDITOR where on the current output device the physical tab stops have been set. PTABSET has no affect on input; hitting the tab key (as opposed to TAB symbol) will put a physical TAB character in the file, rather than actually create spaces. PTAB1-8 must be column numbers, in increasing order. This command has no direct use; it permits EDITOR to output faster on a device with a physical tab mechanism.

QUIT

The QUIT command tells EDITOR you are done editing and want to return to PRIMOS level.

If you have created/modified a file during the session, EDITOR will respond with:

FILE MODIFIED, OK TO QUIT?

WARNING

This message asks whether EDITOR may throw away the work file.

A YES (or Y, YE, O, OK, or null line (CR)) response will QUIT you; you will get back an upper-case OK response, meaning you're at PRIMOS level. Any response except YES, YE, Y, O, or OK will provoke a PLEASE FILE (see FILE); doing a FILE, with or without a filename (depending on the circumstances), will automatically QUIT you.

If you did not create or modify a file, saying QUIT automatically returns you to PRIMOS with an OK.

```
OK, ed poem
GO
EDIT
print 2
.NULL.
'Twas brillig, and the slithy toves
quit
```

```
OK, ed poem
GO
EDIT
delete 3
quit
FILE MODIFIED, OK TO QUIT? yes
```

```
OK, ed poem
GO
EDIT
delete 3
quit
FILE MODIFIED, OK TO QUIT? no
PLEASE FILE
?
file short.poem
```

### RETYPE string

The RETYPE command deletes the current line and replaces it with the text in string.

Exactly one space must be between the command word RETYPE and string.

```
print
Did gyre and gimbol in the wabe.
retype Did a wild watusi in the wabe.
print
Did a wild watusi in the wabe.
```

The string is terminated by either a semicolon (;) or a carriage-return (CR).



Note

RETYPE followed by one space before a carriage-return will act as a DELETE, erasing the current line and leaving the pointer at the NULL line. RETYPE followed immediately by a Carriage-Return yields the error message: BAD R.

SYMBOL name character

The SYMBOL command changes the current character value of a special Symbol to the new value. The Special Symbols and their initial (System default) values are:

<u>Special Symbol Name</u>	<u>Character</u>
KILL	?
ERASE	"
WILD	!
BLANKS	#
TAB	\
ESCAPE	↑
CPROMPT	\$
DPROMPT	&
SEMICOLON	;

You cannot use any of the following as a Special Symbol Character:

- Multiple characters (e.g., ABC)
- comma (,)
- space ( )
- asterisk (\*)
- any character currently in use as a Special Symbol  
except the CPROMPT or DPROMPT

You can check the current value of your symbols with the PSYMBOL command.

```

psymbol
KILL      ?
ERASE     "
BLANKS    #
WILD      !
TAB       \
ESCAPE    ^
CPROMP    $
DPROMP    &
COUNT    @
SEMICO    ;
symbol kill {
symbol tab *
BAD SYMBOL
symbol tab ?
symbol escape %
symbol wild ~
psymbol
KILL      {
ERASE     "
BLANKS    #
WILD      ~
TAB       ?
ESCAPE    %
CPROMP    $
DPROMP    &
COUNT    @
SEMICO    ;

```

EDITOR automatically resets the symbols back to their default values whenever you QUIT or FILE.

The Kill and Erase characters can also be changed via the KILL and ERASE commands.

#### Note

The SYMBOL command is very helpful if you want to use semicolons, quotes, or the question mark in your text.

TABSET tab-1 tab-2... tab-8

The TABSET command sets "logical" tab stops. When EDITOR sees a tab character (the backslash, unless you've changed it, via the SYMBOL command) in INPUT Mode, or in the text of an APPEND, INSERT, OVERLAY, or RETYPE command, it inserts spaces until it reaches the first column in the line where a logical tab has been set. Default logical tab stops are in columns 6, 12, and 30.

```

insert The\use\of\tabs
print
The use of tabs
tabset 10 20 30 40
insert Set\New\Tab\Stops\Blithely
print
Set      New      Tab      Stops      Blithely

```

TOP

The TOP command repositions the pointer at the Top of the file, just above the first line of text. The contents of the Current Line is .NULL. An INSERT after TOP puts text above the first line of text.

It is often a good idea to TOP before making a FIND, a LOCATE, or a multi-line CHANGE, or Modify.

```

top
print
.NULL.
print2
.NULL.
'Twas brillig, and the slithey toves
top
insert \JABBERWOCKY
insert\" \\by Lewis Carroll
top,print 5
.NULL.
      JABBERWOCKY
      by Lewis Carroll
'Twas brillig, and the slithey toves
Did gyre and gimbol in the wabe.

```

UNLOAD filename [n]

The UNLOAD command copies n lines beginning at the current line from the file begin EDITed into a new file named filename. If n is 0 or omitted, it is assumed to be 1. A negative value for n UNLOADS the preceding n-1 lines and the current line, in the correct order (so, for n, 0=1=-1).

The last line UNLOADed is the new current line.

print  
'Twas brillig, and the slithy toves  
dunload stanza 4  
file

OK, slist stanza  
GO  
'Twas brillig, and the slithy toves  
Did gyre and gimbol in the wabe.  
All mimsey were the borogoves  
And the mome raths outgrabe.  
  
OK,

#### Note

Make sure no file named filename previously exists, or,  
if one does, that you don't want it - EDITOR will delete  
your old file to write the new one.

UNLOAD filename TO string

The UNLOAD TO command copies lines in the work file into a new file named filename; lines are copied starting with the current line and continuing down until a line containing string is found, or until BOTTOM is reached.

dunload stanza to The  
The jaws that bite, the claws that snatch--  
file

OK, slist stanza  
GO  
'Twas brillig, and the slithy toves  
Did gyre and gimbol in the wabe.  
All mimsey were the borogoves  
And the mome raths outgrabe.

"Beware the Jabberwock, my son!

Remember: Don't specify a filename currently in use unless you want the old file wiped out. There must be a blank space between TO and string.

VERIFY

The VERIFY command causes verification output -- i.e., automatic printing of the new current line -- whenever any of the following commands are given:

```

APPEND
CHANGE
FIND
GMODIFY
LOCATE
MODIFY
NEXT
NFIND
OVERLAY
POINT

```

Verification Output can be suppressed via the BRIEF command. VERIFY is the default mode.

If the OUTPUT (DISPLAY) command has been given, verification output goes to the remote display.

WHERE

The WHERE command prints the current line number.

```

where
LINE      1
next3
All mimsey were the borogoves
where
LINE      3
next5
Beware the Jub-Jub bird, my son,
where
LINE      8
next-12
TOP
where
LINE      1
point 7
The jaws that bite, the claws that snatch--
where
LINE      7

```

Note

Line numbers are not part of the actual file; EDITOR maintains them for reference.

XEQ buffer

The XEQ command executes the contents of buffer as a command line.

The possible buffers are:

```

INLIN
EDLIN (current line)
STR.1
.
.
.
STR.n

```

It is possible to nest XEQ commands so that commands in string buffers can be executed as subroutines. A n+1-deep stack is maintained to allow the maximum proper nesting of XEQ commands in the n string buffers. Recursive XEQ's will continue to work as always but the stacking still allows "returning" should the recursion stop. In case of an error, BOTTOM or TOP, XEQing stops and the stack is cleared.

Note that the "\*" command causes execution to start at the beginning of the CURRENT command line. All the rules for nesting \*'s apply, but \*'s may be used in an XEQ sequence so long as only 1 \* is outstanding at any time. Thus, the following command sequence:

```

EDLIN:  X STRC,X STRA,T,P24          (Typed command)
STRA:   I @;*5,X STRB
STRB:   I LAST LIN;*5
STRC:   MODE COUNT

```

will cause the following to be inserted into the file, then cause the first 24 lines of the file to be printed:

```

00001
00002
00003
00004
00005
LAST LINE
LAST LINE
LAST LINE
LAST LINE
LAST LINE

```

\* [n]

The Repeat (\*) causes the previous command string to be repeated n times, and/or until TOP or BOTTOM is reached.

Example:

F /;D;N;\*10

deletes the next 10 lines which begin with /.

```
print
a rose
append is a rose;*2
a rose is a rose
a rose is a rose is a rose
print
a rose is a rose is a rose
insert POST NO BILLS; append or BERTS;*5
POST NO BILLS or BERTS
POST NO BILLS or BERTS
POST NO BILLS or BERTS
POST NO BILLS or BERTS
POST NO BILLS or BERTS
```

## SECTION 9

## THE RUNOFF REFERENCE SECTION

This section contains complete information on all RUNOFF commands excepting Decimalization. The commands are listed in alphabetic order.

## EXPLANATION OF THE COMMAND FORMAT

A RUNOFF command consists of a period (.) followed by a command word, and possibly one or more parameters, like so:

.COMMAND parameter

The Period

The period signals that this line in the source file contains a command, which is to be obeyed, as opposed to text, which is to be processed. The period (.) must precede all RUNOFF commands that appear in your source file. In RUNOFF Command Mode, the period may be omitted.

Command Words

A command word is a word which specifies an action. You may input command words in either upper or lower case (or a combination). In this manual, the underlining of certain letters (CO) in a command word indicate the minimum acceptable abbreviation of the word which RUNOFF will understand. For example, if the format indicated:

.PARAGRAPH

then any of the following would be acceptable in your source file:

.P, .p, .Para, .parag, .PaRaGrAph, etc.

Parameters

A parameter, in RUNOFF, is either the name of a file, a number specifying a line, column, page number, or number of lines or columns, or a piece of text, depending on the command.

Numerical parameters are represented in the formats by:

the letter i for a general number  
the letter m for a number of spaces  
the letter n for a number of lines.

Some commands do not use parameters, some require them, and in others, they are optional. In this manual, parameters which are enclosed in brackets -- e.g. [filename] -- are optional. If you omit them, RUNOFF will use the appropriate default value.



The NULL command (period carriage-return, or just carriage-return) tells RUNOFF to begin (or continue) processing the source text file.

.\*comment

The asterisk (\*) command indicates a comment line. It signals RUNOFF to not process this text line in any way.

.\*This file produces the monthly report.

The Comment command, permits you to insert explanatory comments into your source text files for later reference.

."+text. . .

The + (plus sign) command inserts the subsequent text on this line verbatim (i.e., exactly as it appears in the source file) into the output file. If previous lines were FILLed or ADJUSTed, any mode changes implied by the text on this line apply only to this line. The + command causes an implicit BREAK.

when suddenly a flashing sign caught their eyes:

.\*T\*H\*E\*\*E\*N\*D\*\*I\*S\*\*N\*E\*A\*R\*!!\*

Jody caught Sam by the elbow. "Quick!" she said. "Duck in the alley!"

when suddenly a flashing sign caught their eyes:

\*T\*H\*E\*\*E\*N\*D\*\*I\*S\*\*N\*E\*A\*R\*!!\*

Jody caught Sam by the elbow. "Quick!" she said. "Duck in the alley!"

.>text. . .

The > (right-angle bracket) command inserts subsequent text on this line verbatim and centered between the left and right margins. Any mode changes implied by the text apply only to this line. For example, not FILLing or ADJUSTing, when surrounding lines are both. The > command causes an implicit BREAK.

.>MENU

.>Soup du Jour

.>Pate de Foie Gras Vielle

.>Knadlach

.>Cote de boeuf Londres

.>Fromage

MENU

Soup du Jour

Pate de Foie Gras Vielle

Knadlach

Cote de boeuf Londres

Fromage

./left-text/center-text/right-text/

Apportion the pieces of text as left-justified, centered, and right-justified portions of text. You may omit any of the text portions, but must still give all four delimiters. The slash (/) is the only permissible delimiter here. This means the / may not be used as a text character.

./Left-wing/Middle-of-Road/Right-wing/

Left-wing

Middle-of-Road

Right-wing

.ADJUST

The ADJUST command tells RUNOFF to fill each line with words and adjust the spacing between words until each line is right-justified. ADJUST causes an implicit BREAK.

The ADJUST command takes priority over any previous mode command. ADJUST is the default state.

"Get thee hence, Sir Chilblains," she cried."Get thee hence!" Then she lifted her parasol, balanced on her left foot, and threw the precipititious missile at the traitor knight.

"Get thee hence, Sir Chilblains," she cried.  
"Get thee hence!" Then she lifted her  
parasol, balanced on her left foot, and  
threw the precipititious missile at the  
traitor knight.

.BLANK character

The BLANK command resets the value of the Blank Character. Each Blank character indicates a required blank in the source file and indicates words which must not be broken among lines; these blanks will then be neither suppressed nor padded during FILLing and ADJUSTing.

.blank &  
On November&12,&1923, noted author H.G.&Wells visited New&York&City,  
and travelled by boat to Staten&Island.

On November 12, 1923, noted author H.G. Wells  
visited New York City, and travelled by boat  
to Staten Island.

The default character for Blank is CONTROL-@.

.BMARGIN n

The BMARGIN command sets the bottom margin to n lines. If n is zero or omitted, the bottom margin is reset to the default of 5 lines. If necessary, RUNOFF will recalculate footer placement.

BMARGIN causes both an implicit BREAK and EJECT.

.BREAK

The BREAK command signals the end of a paragraph. RUNOFF will stop FILLing the current output line, and not ADJUST it.

Many commands cause an implicit BREAK, i.e., it is as if you said BREAK after them.

draws forth a choice morsel from a cupboard to regale a  
favorite, that this plague will give his history a  
most agreeable variety.

.break

In like manner did my heart leap within me, when  
I came to the dolorous dilemma of Fort Goed Hoop, which

draws forth a choice morsel from a cupboard to  
regale a favorite, that this plague will give his  
history a most agreeable variety.

In like manner did my heart leap within me, when I  
came to the dolorous dilemma of Fort Goed Hoop,  
which

.CMARGIN m

The CMARGIN command sets the inter-column spacing to m spaces wide. This command cannot be used if you have only one text column at the time. If m is zero or omitted, the column margin(s) is reset to its default value of 5 spaces.

The CMARGIN command causes an implicit BREAK and EJECT.

.COLUMNS i

The COLUMNS command sets the number of text columns on a page to i columns. This command causes an implicit BREAK and EJECT. The default number of columns is 1.

'Twas brillig, and the slithy  
 toves Did gyre and gimbol in the  
 wabe. All mimsey were the  
 borogoves And the mome raths  
 outgrabe.

He took his vorpal sword in hand,  
 Long time the manxome foe he  
 sought; And rested he by the  
 tum-tum tree And stood awhile in  
 thought.

### .DEFINE symbol-name value

The DEFINE command tells RUNOFF to find every use of the specified symbol-name enclosed in symbol characters, and replace it with the indicated value. For example, the command DEFINE Name Arthur Trent tells RUNOFF to replace every occurrence of the word %Name% by Arthur Trent when processing the source file into the output file.

The symbol may be up to six letters in length and may not contain any imbedded commas, parentheses or spaces. You may use two-digit numbers as symbol-names, e.g., 03, 15, etc., but not single-digit numbers.

A value may be up to thirty characters in length and may contain any characters.

Up to sixty (60) symbols may be defined at any one time; if you previously defined symbols which you are not using, and need to now define more new ones, you can UNDEFINE some or all of the old symbols. (See UNDEFINE). Also, you may re-DEFINE an existing symbol; the new definition will replace the old one.

```
.define US King Arthur & His Knights
.define THEM Robin Hood & Ye Merrie Men
.>%US%
.>Do Hereby Challenge
.>%THEM%
.>To A Duel Of Weaponry and Skill
```

```
King Arthur & His Knights
Do Hereby Challenge
Robin Hood & Ye Merrie Men
To A Duel Of Weaponry and Skill
```

### .EFOOTER/left-text/center-text/right-text

The EFOOTER command sets up the footer on even-numbered pages. Any character may be used as a delimiter. The delimiters (the slash, in above format) define the contents of the left, center, and right portions of the footer; their presence is required even if there is no text in a given portion, as the following examples demonstrate:

```
.EFOOTER/left-test///
.EFOOTER@@center-text@@
.EFOOTER/left-text//right-text/
.EFOOTER***right-text*
```

.EHEADER/left-text/center-text/right-text/

The EHEADER command sets up the header on even-numbered pages. Any character may be used as a delimiter. The delimiters (slashes in above format) define the contents of the left, center and right portions of the header; their presence is required even if there is no text in a given portion, as the following examples demonstrate:

```
.EHEADER/left-text///
.EHEADER@@center-text@@
.EHEADER/left-text//right-text/
.EHEADER***right-text*
```

.EJECT

The EJECT command makes the next text input go on a new page. It does not mean "physically eject page of paper from the printer". EJECT will have the same effect of beginning a new page even if you have more than one text column on the page at the time.

Certain commands,

```
BMARGIN
COLUMNS
CMARGIN
LENGTH
QUIT
SMARGIN
TMARGIN
WIDTH
```

cause an EJECT implicitly when given; this prevents pages from having contrasting text formats even if you want it.

.ERASE character

The ERASE command redefines the RUNOFF Erase character, which is usable only in RUNOFF Command Mode.

ERASE changes the current value of the Erase Character to the specified character. (The system default value is the double-quote ("). You cannot give the ERASE command in your input file.

Once you have changed the Erase Character, it is not possible to reset it back to its original default value during this RUNOFF session. PRIMOS will, of course, automatically reset the Erase Character when you leave RUNOFF.

.ERRGO

The ERRGO command suppresses the printing of Error Messages and prompts at the terminal when RUNOFF is processing the source input file. Instead, RUNOFF will process the entire file, complete with the results of any errors. (This is particularly important when running RUNOFF as a phantom user.)

NERRGO - print error messages and wait for prompt - is the default.

.FILE [filename]

The FILE command specifies the name of the output file which your source text file is processed into. If you do not specify a filename at the time you give the command, RUNOFF will ask you for one when it begins processing with this message:

ENTER OUTPUT FILE TREENAME.

If you give a filename which already exists in your current (or specified) UFD, RUNOFF will inquire:

OK TO DELETE OLD filename?

If you respond YES (or Y, OK, YE), RUNOFF will delete the existing file by that name, and assign the name to the new output file. If you respond any other way, RUNOFF will ask:

NEW NAME:

to which you should give a filename not currently in use.

```
$  
PROCESSING...  
ENTER OUTPUT FILE TREENAME: *memo  
OK TO DELETE OLD *RR.8? no  
NEW NAME: *new.memo
```

The FILE command can be used at any point in your source input file; if you want to process text into a different file halfway through the source, give the FILE command plus a new output filename. RUNOFF will EJECT before processing any output to this new file.

.FILL

Enter FILL and NADJUST Mode. Text will be filled, but not adjusted. This is true even if you were in ADJUST - the FILL command overrides any previous mode. You must specify ADJUST to enter ADJUST Mode. This command causes a BREAK.

"Get thee hence, Sir Chilblains," she cried. "Get thee hence!" Then she lifted her parasol, balanced on her left foot, and through the precipititious missile at the traitor knight.

.FLOAT filename

Operates in the same manner as PICTURE, but inserts the contents of an external file rather than simply leaving space. Filename is the name of an external file containing text, a table, or illustration captions.

The external file may contain any legal combination of text and RUNOFF commands, but must be terminated by a RETURN statement.

You have the option of beginning the external file with a size-control line, in the form:

. \*n

where n is the number of lines in the file (including any FLOAT, INSERT, or PICTURE commands). Thus if the file is shorter than one page and will fit on the page currently being processed, it will be inserted immediately following the current line. If larger, the current page will be completed, and the external file will be inserted starting at the top of the next page.

If the size-control line is omitted, RUNOFF assumes that the FLOAT file is one page or more in length, and inserts it starting at the top of the next page.

Parameters may be "passed" to the FLOAT file in the same way as with the INSERT command (See INSERT).

FLOAT commands may be nested up to 10 levels.

.FOOTER /left-text/center-text/right-text/

The FOOTER command sets up the Footer on all pages. Any character may be used as the delimiter. The delimiters (slashes in the format) define the contents of the left, center, and right portions of the Footer. Their presence is required even if there is no text in a given portion, as the following examples demonstrate:

```
.FOOTER /left-text///
.FOOTER @@center-text@@
.FOOTER /left-text//right-text/
. ***right-text*
```

.FROM i

The FROM command defines the page number of the first page in the processed output file which you want to appear as actual text. This number is the page number, as printed by the # sign, not the sequential page number. The sequential page number is the total page count; whereas you can reset i at any time, to any value.) If i is zero, or the command is not given, RUNOFF will start printing at page 1.

RUNOFF will be able to check for the page number, even if you are not explicitly printing it on the page.

The FROM and TO commands are particularly useful when you want to examine only a few pages of processed output from a long source.

.HEADER /left-text/center-text/right-text/

The HEADER command sets up the Header on all pages. Any character may be used as a delimiter. The delimiters (slashes (/), in the format) define the contents of the left, center and right portions of the Header; their presence is required even if there is no text in a given portion, as the following examples demonstrate:

```
.HEADER /left-text///
.HEADER @@center-text@@
.HEADER /left-text//right-text/
.HEADER ***right-text*
```

.HYPHEN character

The HYPHEN command defines the value of RUNOFF's Phantom Hyphen as character. The Phantom Hyphen is a character which, when inserted in a word within your source text file, acts as a signal to RUNOFF that it may, if necessary, hyphenate this word at this point when processing it. The default value for the RUNOFF Phantom Hyphen is the RUBOUT Key (or its octal value of †377).

The Phantom Hyphen itself does not appear in the processed output in either case; for example, if the word

antidisestab.RUBOUT.lishment

would run over the right margin, RUNOFF would process the work like so:

```
text text text text antidisestab-
lishment text text text...
```

But, if the word occurs soon enough in the output line so that it would all fit, RUNOFF would process it as:

```
text text antidesestablishment text
text text text
```



.INDENT *m*

Indent the left margin *m* rightwards (inwards) spaces relative to the current left margin.

If *m* is 0, 5, or omitted, RUNOFF will INDENT 5. Negative values are not permitted; use UNDENT to move the indentation back.

.INDEX *string*

The INDEX command makes an entry into the index file defined by the IXFILE command. The entire *string*, up to but not including the Carriage Return, will have the current page number appended to it and be written to the index file (see .IXFILE). The .INDEX command should immediately follow the reference being indexed to insure that the proper page number is used. If the .INDEX command immediately follows a page EJECT due to putting the last word on the previous page, the previous page number will be used.

.INSERT *filename* [(0,1,...9)]

The INSERT command processes and inserts a new input file into the output file. This file may contain both text and RUNOFF commands (including further INSERT commands). This command is used to insert alternate files without losing the initial (or previous) input file(s). Inserts may be nested up to 13 levels deep.

Notes on parameters

1. You may pass up to 10 symbol values to the file to be inserted. RUNOFF will automatically assign the names %0%, %1%..., %9% to 10 values contained in the INSERT command.
2. The names may be up to six characters each.

Each level of insert file has its own definition of parameters, but parameters can be "passed through" multiple levels by using commands such as .INSERT *filename* (%2%, P2, %0%). If a parameter is null (as in P1, P2), any reference to it will be treated as an undefined symbol and deleted. If multiple parameters are concatenated to form a variable length string, a reference of the form: %1%%2%%3% can be used to reference up to 16 characters. However, if less than 13 are needed as in:

```
.INSERT filename (NAME, Abcdef,ghi
```

The following reference

```
%0% = %1%%2%%3%
```

will be treated as:

```
NAME - Abcdefghi.
```

### .IXFILE filename

The IXFILE command defines the name of an index file. If IXFILE is given, RUNOFF will copy all entries flagged by INDEX in the source program, into the filename with page numbers.

#### Note

RUNOFF will neither combine multiple entries nor format the file; this must be done via EDITOR.

If you do not IXFILE, and your RUNOFF source file contains one or more INDEXes, RUNOFF will ask for the name of an index file:

#### INDEX FILE

You may then either specify filename, which becomes the index file, or hit a Carriage Return. The latter tells RUNOFF that you do not want an index or index file; RUNOFF will ignore all subsequent INDEX commands in the file. (This is equivalent to doing a NIXFILE).

If filename already exists, RUNOFF will ask if it is OK to delete the old file filename; if you answer NO, RUNOFF will then request a new filename.

### .KILL character

The KILL command defines the RUNOFF Kill Character, which can only be used in RUNOFF Command Mode, to the specified character. This command cannot be used in the source input file.

The default value of the RUNOFF Kill Character is the question mark(?).

If you change this value via the KILL command, you cannot reset it back to this default value during this RUNOFF session - however, PRIMOS will automatically reset the KILL character value when you leave RUNOFF.

### .LENGTH n

Defines physical page length, including top and bottom margins, as n lines. RUNOFF will recalculate placement of headers and footers. The LENGTH command causes an implicit BREAK and EJECT.

The default value of n is 66 lines. (At 6 lines = 1 inch, 66 lines = 11 inches).

.NADJUST

The NADJUST command turns off the adjusting of the right margins of output lines. If you were in ADJUST Mode, giving the NADJUST command is equivalent to saying FILL; if you were already in FILL, NADJUST has no effect.

.NEED n

The NEED command specifies that a block of n printing lines - actual lines of text, not page lines - are needed for a body of text to follow.

If n lines are available on the current page, this command has no effect; if n lines are not available, this command causes a BREAK and EJECT to top of next column or page.

If n is zero or omitted, 1 line is assumed.

.NERRGO

The NERRGO command tells RUNOFF to halt processing and display an error message and prompt whenever it finds a command error in the source text file. RUNOFF will skip a line, insert the erroneous command in the output file as text, skip another line, and return to command mode. Typing a null line (CR only) causes RUNOFF to continue processing where it left off. At the end of the file, RUNOFF will return to PRIMOS via the ER! prompt.

Error messages may be suppressed via the ERRGO command. NERRGO is the default state.

.NFILE

The NFILE command specifies that the processed output should not be put into a file. This command has little use unless you are also giving the TTY command which displays the output at your terminal as it is processed; otherwise, you will have nothing to show for having processed the source file except for the knowledge of whether or not the source file has any command errors in it.

.NFILL

The NFILL command tells RUNOFF:

Do not FILL.  
Do not ADJUST

and causes an implicit BREAK. In NFILL Mode, the tab character is recognized.

NFILL takes priority over any previous mode. NFILL Mode is particularly useful for formatting tables.

"My dear Fortunato, you are luckily met. How remarkably well you are looking today. But I have received a pipe of what passes for Amontillado, and I have my doubts."

"How?" said he. "Amontillado? A pipe? Impossible! And in the middle of the carnival!"

#### .NIXFILE

The NIXFILE command tells RUNOFF that despite the appearance of INDEX commands in your RUNOFF source file, you do not want to generate an index file.

#### .NPARAGRAPH

The NPARAGRAPH command resets the paragraph values to their default of INDENT 0, SKIP 1.

#### Note

NPARAGRAPH does not signal a new paragraph. It just resets these values.

#### .NPAUSE

The NPAUSE turns off the-between-page pause activated by the PAUSE command. NPAUSE is the default.

#### .NPERFORATE

The NPERFORATE command de-activates the perforation marks between pages, as activated by PERFORATE, and EJECTS to the new page.

#### .NTTY

The NTTY command tells RUNOFF not to print processed output at the terminal. NTTY is the default.

.OFOOTER /left-text/center-text/right-text/

The OFOOTER command sets up the Footer on odd-numbered pages. Any character may be used as the delimiter. The delimiters (slashes, in the format) define the contents of the left, center and right portions of the Footer. Their presence is required even if there is no text in a given portion, as the following examples demonstrate:

```
.OFOOTER /left-text///
.OFOOTER @@center-text@@
.OFOOTER /left-text//right-text/
.OFOOTER ***right-text*
```

.OHEADER /left-text/center-text/right-text/

The OHEADER command sets up the Header on odd-numbered pages. Any character may be used as a delimiter. The delimiters (slash (/), in the format) define the contents of the left, center and right portions of the Header; their presence is required even if there is no text in a given portion, as the following examples demonstrate:

```
.OHEADER /left-text///
.OHEADER @@center-text@@
.OHEADER /left-text//right-text/
.OHEADER ***right-text*
```

.PAGEN i

The PAGEN command specifies a new starting page number. The next page to begin after this command will be numbered i. The number will be inserted wherever the # character is used in a HEADER or FOOTER command. For example:

```
.pagen 5
.HEADER /text/text/Page #/
```

will make the next page have a header with "Page 5".

.PARAGRAPH [m] [n]

The PARAGRAPH command signals the beginning of a new paragraph; RUNOFF does a BREAK in the current output line, and then INDENTS m from the current left margin and SKIPS n lines.

...just in time to see it pop down a large  
rabbit hole under the hedge.

.paragraph 5 1

In another moment down went Alice after it,  
never once considering how in the world she was  
to get out again.

.paragraph

The rabbit hole went straight down like a tunnel  
for some way, and then dipped suddenly down...

...just in time to see it pop down a large rabbit  
hole under the hedge.

In another moment down went Alice after it,  
never once considering how in the world she was to  
get out again.

The rabbit hole went straight down like a  
tunnel for some way, and then dipped suddenly  
down...

If you do not specify m or n, RUNOFF will use the most recently specified values. The default values for m and n are INDENT 0, SKIP 1. The NPARAGRAPH command will reset m and n to these default values without actually starting a new paragraph.

You can also signal a paragraph in FILL or ADJUST Modes by beginning on input line with a space. This is known as implicit paragraphs.

The command PARAGRAPH 0 0 is equivalent to BREAK.

#### Note

Hanging Indents - You can also specify negative indentations with PARAGRAPH, e.g., PARAGRAPH -8 2. This permits you to do negatively-indented text; use the SKIP command to skip lines without starting new paragraphs, like so:

```

.NFILL
,PARAGRAPH -8 2
This paragraph is done with a hanging indent
line two
line three
.PARA
Second paragraph under hanging indent
line one
.SKIP
line two

```

gives:

```

This paragraph is done with a hanging indent
line two
line three

Second paragraph under hanging indent
line one

line two

```

#### .PAUSE

The PAUSE command causes RUNOFF to:

- 1) Pause when each new page of output is ready to be printed.
- 2) Ring the terminal bell.
- 3) Wait until a character is typed at the terminal before printing the new page.

The PAUSE command permits you to:

- On printing terminals, print each page of RUNOFF output on a separate piece of paper.
- On CRT terminals, inspect the output file page by page.

Although any character you type will start the printing output of the new page, you probably want to type a non-printing character to avoid the problem of later erasing this character.

The default is NPAUSE.

Unless you are adjusting paper at a nearby line printer, you should also specify .TTY, to direct output to your terminal as RUNOFF processes your source file. If you do not want to save the resulting output file, also give the .NFILE command.

The value 1 - means "Read character from terminal" - is the default value when PAUSE is used.

### .PERFORATE [n]

The PERFORATE command prints a line of hyphens as perforation marks between pages, on the terminal and the output file.

If a value consisting of a positive integer (e.g., 1, 2, 200, etc.) is used for n, the perforation line will consist only of a hyphen at each end of the line. If you give the PERFORATE command again, with the value of n anything but a positive integer, the regular perforation marks are restored.

The NPERFORATE command turns off the perforation marks.

### .PICTURE n

The PICTURE command reserves n physical page lines (6=1") (as opposed to printing lines) for later insertion of a figure. The rules for this reservation are:

1. If n physical lines remain on the current page, these lines are skipped, if you're in FILL Mode, the current line of output text is completed before the skip occurs - i.e., a break does not occur.
2. If n lines do not remain on the page, RUNOFF will continue to process the output page as normal, and then skips n lines beginning at the top of the next page (or column).
3. If n is larger than the number of available lines (excluding top and bottom margins) on a page, RUNOFF will skip pages and lines until a total of n lines have been skipped.

RUNOFF will keep track of up to ten (10) PICTURE requests at a time, i.e., up to ten requests for which space has not yet been skipped. RUNOFF will attempt to satisfy each PICTURE request as soon as possible, and putting exactly one PICTURE per page and filling the rest of each page with text.

If you specify more PICTURE requests while there are ten outstanding requests, the additional requested space will be added to that of the tenth PICTURE.

### .PURGE

The PURGE command forces immediate satisfaction of all outstanding pictures and floats on the current (and all nested) FLOAT levels. This is necessary when you do not want a nested FLOAT to float beyond the textual end of a higher-level FLOAT.



### .QUIT

The QUIT command exits you from RUNOFF back to PRIMOS command level. If RUNOFF finds a QUIT in the input source file, the current page of output is EJECTED before returning to PRIMOS, which will be indicated by the PRIMOS prompt, OK.

Hitting the Escape (or Break) Key on your terminal will function exactly like typing the QUIT command. You can restart RUNOFF by typing the START command, with or without a filename (i.e., START or START filename).

If you type QUIT while in RUNOFF Command Mode, you may lose the last page of processed output.

### .RBAR [ON]

The RBAR command turns on and off a Revision Bar. A Revision Bar is a vertical line (or bar) printed just to the left of the text, and is used to indicate that the corresponding text has been changed since some previous edition of the document.

The command .RBAR ON turns on the Revision Bar. The bar will appear on all subsequent lines until the RBAR command appears again with any parameter value other than ON - e.g., RBAR OFF, RBAR XXX, RBAR 5. In ADJUST Mode, if the Revision Bar is turned off in the middle of a line, the line will still be marked by a bar.

RBAR will not put a revision bar next to any blank line.

### .RETURN [i]

The RETURN command returns you to previous input (.INSERT) file.

If i is zero or omitted, the current file will be closed if it was actually opened by RUNOFF. If i = 1, the current file will be left open. In all cases, return is to the previous input level. If a .RETURN is encountered in the primary input file, return is to Command Mode. If 1 is used typing a null line or .INSERT with no parameter will cause processing to continue in the original file as if nothing had happened. This can be used to allow dynamic parameter changes during processing from the TTY. If .RETURN is used (i=0) to Command Mode, then resumption is only possible if the file were not explicitly opened by RUNOFF. Otherwise, only .INSERT filename or .INSERT i can be used to resume file processing from a new file. The .INSERT/.RETURN combination is implemented using a file unit stack to insure proper nesting of input files, down to 13 levels. .RETURN always returns 1 level and .INSERT always goes down 1 level. A typical use of these commands for form letters is shown

below:

<u>Commands</u>	<u>CTRL</u>	<u>DUMMY</u>	<u>Name &amp; Addresses</u>
RUNOFF CTRL \$.NULL.			
	.INSERT DUMMY		
		.INSERT NAMES	
		.RETURN	.RETURN 1
		<u>FORM</u>	
	.INSERT FORM	text	
		.INSERT	
			namel
			.RETURN 1

<u>Commands</u>	<u>CONTRL</u>	<u>DUMMY</u>	<u>Name &amp; Addresses</u>
		text .INSERT	address1 .RETURN 1
		text .EJECT .RETURN	
	.INSERT FORM		.
	.		.
	.		.
	.QUIT		.RETURN

The use of DUMMY and the initial .RETURN 1 in the names and addresses file is to open the names file on the proper unit for that level of nesting and leave it open.

#### .RINDENT [*m*]

The RINDENT command indents the right margin, by moving it m spaces to the left of the current right margin. If m is  $\emptyset$  or omitted, the right margin is indented by the default value of 5 spaces. The RUNDENT command de-undents the right margin.

#### .RUNDENT [*m*]

The RUNDENT command de-indent (undents) the right margin, moving it m spaces to the right of the current right margin.

If m is zero or omitted, the right margin is reset to the original right margin as specified by the SMARGIN command.

#### .SKIP [*n*]

Skip n printing lines - i.e., n times line spacing. (If double-spacing, skip 2n lines, etc.)

SKIP causes an implicit BREAK.

These names of virtue, with  
their precepts, were:

.skip 2

.>1. TEMPERANCE.

.skip 1

Eat not to dullness; drink not to elevation.

.skip 2

.>2. SILENCE.

.skip 1

Speak not but what benefit others  
other yourself; avoid trifling conversation.

.skip 2

.>3. ORDER.

.skip 1

Let all your things have their places; let  
each part of your business have its time.

These names of virtue, with their precepts,  
were:

#### 1. TEMPERANCE.

Eat not to dullness; drink not to elevation.

#### 2. SILENCE.

Speak not but what benefit others other  
yourself; avoid trifling conversation.

#### 3. ORDER.

Let all your things have their places; let  
each part of your business have its time.

#### Note

SKIP  $\emptyset$  is equivalent to .BREAK.

#### .SMARGIN[m]

The SMARGIN command resets the current side margins (i.e., left and right) to m spaces from the edges of the page (as defined by WIDTH or its default value).

If m is  $\emptyset$  or omitted, the side margins are set to the default of 7 spaces.

SMARGIN causes an implicit BREAK and/or EJECT.

.SOURCE [n]

The SOURCE command generates a list of line numbers one space to the right of the margin of the output file, each number corresponding to a non-blank text line from the source input file. Line numbers are four digits, enclosed in parentheses.

Each successive line number is n greater than the preceding number; if the value of n is 1 or omitted, the line numbers will increase by 1; if n = 2, by 2, etc. If n = 0, the line numbering is terminated. The SOURCE command may be given in the source input file or in RUNOFF command mode.

Only source lines that generate output text will be numbered; command lines e.g., PARA, .WIDTH, etc., will not appear in processed output, nor will they be numbered. If an input line becomes several lines of processed text, the first of these lines will be numbered.

If you have multi-column output, each column will have a column of corresponding source line numbers.

For files put into your output via INSERT or FLOAT, Source numbering will restart at 1 for the duration of the file; at the end of these files, the line number will be reset to that of the line which followed the INSERT command, or the line after the insertion of a float.

These names of virtue, with their precepts, (0001)  
were:

1. TEMPERANCE. (0004)

Eat not to dullness; drink not to elevation. (0006)

2. SILENCE. (0008)

Speak not but what benefit others other (0010)  
yourself; avoid trifling conversation.

3. ORDER. (0013)

Let all your things have their places; let (0015)  
each part of your business have its time.

.SPACE [n]

The SPACE command sets the spacing mode for printing output lines. n = 1 is single spacing, n = 2 is double spacing, etc. If n = 0, or omitted, the default of single spacing is used. If n is set to a value larger than the number of available lines (not including margins) per page, only 1 line will be printed per page (column).

The SPACE command does an implicit BREAK.

.STOP

The STOP command is a conditional QUIT - if STOP is encountered in RUNOFF Command Mode or in the source input file, it is treated as a QUIT. However, if encountered in an inserted or floated file, it is treated as a .RETURN (with i = 0).

In all cases, an end-of-file (EOF) on an input file is treated exactly like a STOP command.

.SYCHAR character

The SYCHAR command defines the specified character as the delimiter for symbol names - i.e., these characters must enclose symbol-names in the source file. The default value of the SYCHAR character is the percent sign (%).

.TAB character tab-1 tab-2...tab-20

The TAB command devines the current tab character and stops for RUNOFF. The tab symbol is set by character, which can be any character not currently defined by EDITOR or RUNOFF. (If the character has a special meaning in EDITOR, it will be processed when you input it, and never actually appear in the source file.) RUNOFF has no default tab character.

The AT sign (@) is commonly used as the RUNOFF Tab symbol.

## WARNING

Remember that the backslash (\) is recognized as a tab symbol only by EDITOR, and is converted to tab spaces immediately upon input - spaces which RUNOFF will suppress. In the TAB command, there must be a space between the command word, the character, and each tab setting. If for some reason, the character is omitted or no stops are set, tabbing will not occur. Tabs must be set going from left to right; if you do not do this, the message ILLEGAL COMMAND will print.

In NFILL Mode, you may sue the tab symbol anywhere on a line, and it will be interpreted correctly. However, in ADJUST and FILL Modes, tab symbols will ge interpreted correctly only if the input line starts with a TAB symbol. There are no default tab stops; there is no pre-defined tab symbol.

#### .TMARGIN [n]

The TMARGIN command sets the top margin of the page to n lines down from the top. The placement of headers and footers will be recalculated, if necessary. If the value of n is zero, 7, or omitted, the top margin will be reset to the default value of seven lines.

This TMARGIN command causes an implicit BREAK and EJECT.

#### .TO i

The TO command defines the page number of the last page in the output file which you want. RUNOFF will stop processing the source file as soon as page i is filled. The page number is the number initialized by the PAGEN command, as opposed to the sequential page number. (The sequential page number is the total page count; whereas you can reset i at any time, to any value.) The # symbol is replaced by the page number during processing; but RUNOFF will know a page number even if there is no # on the page's format.

If the TO command is not given, RUNOFF will process the entire source file. The FROM and TO commands are particularly useful when you only want to examine a few processed pages of a very long source file.

Although these commands are usually entered in RUNOFF Command Mode, they may be located anywhere in the source file.

#### .TTY

The TTY command causes RUNOFF output to be printed at the user's terminal.

#### .UNDEFINE [symbol]

The UNDEFINE command undefines the value of the specified symbol, and removes this symbol from the symbol table. If a particular symbol is not specified, i.e., the command UNDEFINE is given - the entire symbol table will be cleared.

#### .UNDENT [m]

The UNDENT command undents - i.e., moves the current left margin - by m spaces to the left. If the value of m is zero or omitted, the left margin will be reset to the original left margin specified by the side margin command, SMARGIN, (or its default value of 7 spaces, if not explicitly given).

.WIDOW [n]

The WIDOW command prevents you from having widows of up to n lines on your output pages. A widow is one or more lines of text at the bottom of a page which are separated from the rest of the text by blank lines. The WIDOW command tells RUNOFF to check for widows at the bottom of all subsequent pages. If RUNOFF sees a line skip with n+1 lines of the bottom margin, it does an EJECT after that skip.

WIDOW-checking is turned off by giving the command WIDOW with the value of n either omitted or zero. WIDOW-checking should be turned off prior to processing tables and diagrams, and turned on afterwards as desired.

The default value for n is zero.

.WIDTH [m]

The WIDTH command defines the physical page width, including both left and right margins. The default page width is 85 spaces; the maximum allowable in RUNOFF is 170 spaces.

(At 10 spaces = 1 inch; 85 spaces = 8-1/2 inches; 170 spaces = 17 inches.)

The WIDTH command causes an implicit BREAK and EJECT.



## INDEXES

This index consists of four parts:

- A general subject index
- A cross-reference of the PRIMOS commands described in this guide
- A cross-reference of the EDITOR commands
- A cross-reference of the RUNOFF commands

# MAIN INDEX TO THE NEW USER'S GUIDE

" , ENTERING WITH EDITOR	3-21	BANNER	8-17
#	5-5	BASIC EDITOR COMMANDS	3-8
#	9-14	BEGINNING A TEXT LINE WITH A PERIOD IN RUNOFF	6-9
%	6-8	BLANK CHARACTER	6-7
*	9-2	BLOCKS OF TEXT, SAVING, IN RUNOFF	6-12
+	9-2	BOTTOM MARGIN	6-2
.NULL.	3-6	BRACES	6-10
1/2	4-4	BRACES, ON DIABLO PRINTER	6-11
24-HOUR TIME	2-5	BRACKETS	3-2
>	9-2	BUSY PHONE LINES	2-6
?, ENTERING WITH EDITOR	3-21	CANCELLING A SPOOL REQUEST	4-7
ACOUSTIC COUPLER	2-3	CARRIAGE-RETURN	2-2
ACOUSTICAL COUPLER	2-6	CASE	1-11
ACOUSTICAL COUPLERS	1-3	CASE SWITCH	1-11
ADJUST MODE	5-7	CATHODE RAY TUBE TERMINALS	1-9
ADJUST MODE	5-8	CENTRAL PROCESSING UNIT	2-8
APPORTIONING	5-11	CHARACTER SYMBOL COMMANDS, RUNOFF	5-5
ARS LONGA	SEE VITA BREVIS	CHARACTER	1-8
ART INSERTION	9-17	CHARACTER	1-9
ART, SPACE FOR	6-13	CHARACTER	3-2
ASTERISK	9-2	CHARACTER	3-2
ATTACHING TO A SUB-UFD	4-4	CNTRL	1-11
ATTACHING TO ANOTHER UFD	4-1	COLUMN HEADER DISPLAY, EDITOR	8-17
ATTACHING TO ANOTHER UFD	4-4	COLUMNS OF TEXT, NUMBER OF	6-3
BACKSLASH	1-11	COMMAND	1-6
BACKSLASH	3-21		
BACKSLASH CHARACTER	1-11		

# MAIN INDEX TO THE NEW USER'S GUIDE

COMMAND CONVENTIONS, RUNOFF'S 5-4	COPYING A FILE 3-20
COMMAND ERROR MESSAGES, RUNOFF, SUPPRESSING	CORRECTING RUNOFF FILES 5-19
COMMAND ERRORS, RUNOFF 5-20	COUNTER, INCREMENTING 8-16
COMMAND FORMAT 1-7	CPU TIME 2-8
COMMAND FORMAT, EDITOR 3-1	CREATING A NEW FILE WITH EDITOR 3-19
COMMAND FORMAT, EDITOR 8-1	CREATING THE SOURCE FILE 5-2
COMMAND FORMAT, RUNOFF 9-1	CRT TERMINALS 1-9
COMMAND MODE, RUNOFF'S 5-18	CURRENT DIRECTORY 4-2
COMMAND WORD 5-4	CURRENT LINE 3-6
COMMAND WORDS 1-6	CURRENT LINE NUMBER, GETTING 8-33
COMMAND WORDS 3-1	CURRENT UFD 4-1
COMMAND WORDS 8-1	CURRENT UFD 4-4
COMMAND WORDS 9-1	DATA 1-6
COMMAND, 5-4	DATA 1-7
COMMANDS, ILLEGAL 5-20	DECIMALIZATION 7-1
COMMANDS, UNRECOGNIZED 5-20	DEFAULT PAGE, RUNOFF'S 5-5
COMMON SOURCE OF ERRORS 5-20	DEFAULTS 1-7
CONNECTING COMPUTER TO TERMINAL 2-3	DEFAULTS 5-5
CONTROL COMMANDS, EDITOR'S 3-26	DELETING FILES AND SUB-UFDS 4-8
CONTROL KEY 1-11	DELIMITER, SYMBOL-NAME 6-8
CONVENTION IN THE EXAMPLES 1-9	DIABLO PRINTER, INPUTTING BRACES 6-11
CONVENTIONS 1-7	DIALING IN 2-4
CONVENTIONS IN EDITOR 3-1	DIALOGUE 1-6
CONVENTIONS, POLITICAL SEE HISTORY BOOK	DISPLAYING A FILE ON A TERMINAL 4-5

# MAIN INDEX TO THE NEW USER'S GUIDE

DISPLAYING RUNOFF OUTPUT ON TERMINAL 5-14	EDITOR'S MODE COMMANDS 3-27
DOING A BREAK 5-11	EDITOR'S OTHER COMMANDS 3-25
DOING AN IMPLICIT BREAK 5-11	EDITOR'S POINTER-MOVING COMMANDS 3-10
DOUBLE-QUOTE CHARACTER 2-2	EDITOR'S PRINTING COMMANDS 3-9
DOWN 2-7	EDITOR'S STRING-FINDING COMMANDS 3-12
ED COMMAND 3-3	EDITOR'S SYMBOL-CHANGING COMMANDS 3-27
ED COMMAND PLUS FILENAME 3-4	EDITOR'S TAB CHARACTER 3-21
EDIT MODE 3-3	EDITOR'S TAB STOPS 3-21
EDIT MODE 3-5	EDITOR'S VALUE-SETTING COMMANDS 3-28
EDIT MODE 3-8	EDITOR, CONVENTIONS 3-1
EDIT MODE, GIVING COMMANDS IN 3-7	EDITOR, GENERAL INFORMATION 3-25
EDITING AN EXISTING FILE 3-4	EJECTING TO NEW PAGE 5-1
EDITOR COMMAND FORMAT 3-1	ENDING AN EDITOR SESSION 3-17
EDITOR MODE PROMPTS 8-18	ENTERING EDITOR 3-3
EDITOR SAMPLE SESSION 3-23	ENTERING SEMICOLONS IN INPUT MODE 3-5
EDITOR SESSION, ENDING AN 3-17	ENTERING TEXT IN INPUT MODE 3-4
EDITOR WORK FILE 3-3	ERASE CHARACTER 2-2
EDITOR'S COMAND FORMAT 8-1	ERASE CHARACTER 3-1
EDITOR'S CONTRO COMMANDS 3-26	ERASE CHARACTER, RUNOFF'S 6-7
EDITOR'S ERASE CHARACTER 3-1	ERROR MESSAGE 2-2
EDITOR'S INPUT/OUTPUT COMMANDS 3-28	ERROR MESSAGES 1-6, ALSO SEE INDEX OF SYSTEM MESSAGES
EDITOR'S KILL CHARACTER 3-1	ERROR MESSAGES 1-7
EDITOR'S LINE-CHANGING COMMANDS 3-26	ERROR MESSAGES, RUNOFF,
EDITOR'S LINE-CHANGING COMMMANDS 3-13	

# MAIN INDEX TO THE NEW USER'S GUIDE

SUPPRESSING	6-7	GENERATING A TABLE OF CONTENTS	7-6
ERRORS, COMMAND, RUNOFF	5-20	GETTING READY TO LOG IN	2-2
ERRORS, COMMAND, RUNOFF	5-21	GIVING COMMANDS IN EDIT MODE	3-7
ERRORS, COMMON SOURCE OF	5-20	GIVING LOGOUT COMMAND FROM OTHER UFD	4-1
ESSENTIALS OF EDITOR	3-1	HANGING INDENTS	9-15
EXAMPLES, CONVENTIONS	1-9	HARD COPY	5-14
EXISTING FILENAME	3-19	HARD COPY TERMINAL	4-6
FEATURES, RUNOFF'S	5-5	HARD-COPY TERMINALS	1-9
FIGURE INSERTION	9-17	HEADERS, EVEN AND ODD	6-4
FILE LOADING AND UNLOADING		HI--I'M PRIMOS	2-1
COMMANDS, EDITOR'S	3-25, 26	HIGH-SPEED TAPE READER	8-13
FILENAME	1-8	HITTING THE CARRIAGE-RETURN	2-2
FILENAME	1-9	HOW COMPUTERS THINK	1-5
FILENAME	3-2	HOW EDITOR WORKS ON ITS FILE	3-6
FILENAME, OUTPUT, RUNOFF	6-5	HOW RUNOFF PROCESSES TEXT	5-8, 9
FILENAMES, RULES FOR MAKING	3-20	HOW TO COPY A FILE	3-20
FILES	1-6	HOW TO GET YOUR UFD	2-1
FILES, DELETING	4-8	HYPHEN, PHANTOM	6-8
FILES, RENAMING	4-8	I	5-4
FILL MODE	5-7	I/O TIME	2-8
FILL MODE	5-8	IDENTICAL FILENAMES, CANNOT HAVE	3-19
FINDING OUT WHAT'S IN YOUR UFD	2-7	ILLEGAL COMMANDS	5-20
FOOTERS, EVEN AND ODD	6-4	IMPLICIT BREAK	5-11
G (FOR GENERAL)	3-15		
G (FOR GENERAL)	8-5		
GENERAL INFORMATION ABOUT EDITOR	3-25		

# MAIN INDEX TO THE NEW USER'S GUIDE

IMPLICIT PARAGRAPHING IN RUNOFF 5-13	KILL CHARACTER 3-1
INDENTATION 6-3	KILL CHARACTER, RUNOFF'S 6-7
INDENTING AND UNINDENTING 6-3	LAYOUT OF TERMINAL KEYBOARD 1-10
INDENTS, HANGING 9-15	LINE MODE 1-10
INDEXING 6-13	LINE MODE 2-3
INDEXING COMMANDS, RUNOFF'S 6-1	LINE NUMBER, CURRENT 8-33
INDICATOR LIGHT 1-10	LINE NUMBERS 3-11
INITIAL TEXT PERIOD 6-9	LINE NUMBERS 3-7
INPUT 1-6	LINE NUMBERS 9-22
INPUT AND EDIT MODE 3-3	LINE-CHANGING COMMANDS, EDITOR'S 3-26
INPUT FILE, SPECIFYING 5-18	LINE-CHANGING COMMANDS, EDITOR'S 3-13
INPUT MODE 3-3	LINE-FORMATTING COMMANDS 5-7
INPUT MODE 3-8	LINE-FORMATTING COMMANDS, RUNOFF'S 5-4
INPUT/OUTPUT COMMANDS, EDITOR'S 3-28	LINE-ORIENTED SYSTEM 3-6
INPUTTING A NEW FILE 3-3	LINE/LOCAL 1-10
INSERTING FILES, IN RUNOFF 6-13	LINE/LOCAL SWITCH 1-10
INSERTION OF FIGURES 9-17	LINE/LOCAL SWITCH 2-3
INTER-COLUMN MARGIN 6-2	LINES NUMBERS, RUNOFF OUTPUT 6-6
INTERACTIVE DIALOGUE 1-6	LITERAL DOUBLE-QUOTE 3-21
IS TERMINAL IN LINE MODE 1-11	LITERAL QUESTION-MARK 3-21
JOB NUMBER 2-5	LITERAL SYMBOL DELIMITER, INSERTING 6-9
KEYBOARD, TERMINAL, LAYOUT 1-10	LOCAL MODE 1-10
KEYWORDS 1-9	LOCAL MODE 2-6
KILL CHARACTER 2-2	LOGGING IN 2-2

# MAIN INDEX TO THE NEW USER'S GUIDE

LOGGING IN	2-4	NFILL MODE	5-7
LOGGING IN, PROBLEMS	2-5	NFILL MODE	5-8
LOGGING OUT	2-8	NO OUTPUT FILE WANTED FOR RUNOFF	6-5
LOGIN COMMAND	2-4	NULL COMMAND	3-8
LOGIN PLEASE	2-5	NULL COMMAND LINE	3-8
LOGOUT COMMAND	2-8	NULL INPUT LINE	3-5
LOGOUT COMMAND	4-1	NULL INPUT LINE	3-6
LOOKING AT YOUR FILES	4-5	NULL LINE	3-7
LOWER-CASE	1-11	NULL LINES	3-6
M	5-4	NUMBER OF TEXT COLUMNS	6-3
MAKING INDEXES	6-13	NUMBERING RUNOFF OUTPUT LINES	6-6
MAKING SUB-UFDS	4-2	NUMERIC PARAMETERS	1-8
MARGINS	6-2	NUMERIC PARAMETERS	3-2
MAXIMUM DECIMAL HEADING NUMBER	7-2	ON/OFF	1-10
MESSAGES, ERROR, COMMAND, RUNOFF, SUPPRESSING	6-7	ON/OFF	2-3
MODE COMMANDS, EDITOR'S	3-27	ON/OFF SWITCH	1-10
MODE PROMPTS, EDITOR	8-18	ONE-SLASH-TWO	4-4
MONITOR SYSTEM	2-1	OPERATING SYSTEM	2-1
MORE PRIMOS	4-1	OTHER COMMANDS, EDITOR	3-25
MORE RUNOFF	6-1	OTHER REASONS YOU CAN'T LOG IN	2-6
N	1-8	OUTPUT	1-7
N	3-2	OUTPUT COMMANDS, RUNOFF'S	5-4
N	5-4	OUTPUT COMMANDS, RUNOFF'S	6-1
NEGATIVE INDENTATION	9-15	OUTPUT FILE	5-2
NESTED FLOAT COMMANDS	9-8	OUTPUT FILE, RUNOFF	5-2
NEWLINE	3-2		

PDR-3104

# MAIN INDEX TO THE NEW USER'S GUIDE

OUTPUT FILE, RUNOFF, NAMING 5-18	PARENTHESES 1-8
OUTPUT FILENAME, RUNOFF 6-5	PASSING PARAMETERS 9-11
OUTPUT OPTIONS, RUNOFF'S 6-4	PASSING PARAMETERS 9-8
PAGE NUMBER 5-5	PASSWORD 1-3
PAGE NUMBER 9-24	PASSWORD 2-2
PAGE SIZE COMMANDS 6-2	PASSWORDS 2-1
PAGE WIDTH 9-25	PAUSING BETWEEN PAGES OF RUNOFF OUTPUT 6-6
PAGE-FORMATting COMMANDS, RUNOFF'S 5-4	PERCENT SIGN 6-8
PAGE-FORMATting COMMANDS, RUNOFF'S 6-1	PERIOD 9-1
PAGES, SELECTED, PROCESSING 6-5	PERIOD, INITIAL TEXT 6-9
PAPER TAPE READER, HIGH SPEED 8-13	PHANTOM HYPHEN 6-8
PAPER TAPE READER, TELETYPE 8-13	PHANTOM HYPHEN 9-9
PARAGRAPHING IN RUNOFF 5-12	PHANTOM HYPHEN, DEFAULT VALUE 6-11
PARAGRAPHING, IMPLICIT, IN RUNOFF 5-13	PHANTOM HYPHEN, USING TO ENTER LITERAL DOUBLE BRACE 6-11
PARAMETER PASSING 9-10	PHANTOM USERS 6-7
PARAMETER PASSING 9-8	PHONE NUMBER TO COMPUTER 2-4
PARAMETERS 1-6	PHYSICAL PAGE, WIDTH OF 9-25
PARAMETERS 3-2	PHYSICAL TAB STOPS 8-27
PARAMETERS 5-4	PLUS SIGN 9-2
PARAMETERS 8-1	POINTER 3-6
PARAMETERS 9-1	POINTER-MOVING COMMANDS, EDITOR'S 3-10
PARAMETERS, NUMERIC 1-8	POUNDS SIGN 5-5
PARAMETERS, TEXT 1-8	POWER SWITCH FOR TERMINAL 2-3
	PRIMOS 1-3
	PRIMOS 1-5



# MAIN INDEX TO THE NEW USER'S GUIDE

PRIMOS	2-1	RUBOUT KEY	1-11
PRIMOS ERASE CHARACTER	2-2	RUBOUT KEY	6-11
PRIMOS KILL CHARACTER	2-2	RULES FOR DEFINING RUNOFF SYMBOLS	6-9
PRINTING COMMANDS, EDITOR'S	3-9	RULES FOR MAKING FILENAMES	3-20
PROBLEMS IN LOGGING IN	2-5	RUNNING RUNOFF	5-16
PROCESSING SELECTED PAGES ONLY	6-5	RUNOFF'S DEFAULT PAGE	5-5
PROCESSING SEQUENCE, RUNOFF, SUMMARY	5-19	RUNOFF	5-1
PROCESSING THE SOURCE FILE	5-16	RUNOFF COMMAND ERRORS	5-20
PROOFREADING	5-19	RUNOFF COMMAND ERRORS	5-21
PUNCHING OUTPUT TAPES WITH EDITOR	8-27	RUNOFF COMMAND MODE	5-18
QUESTION MARK	1-7	RUNOFF COMMANDS	5-2
REDEFINING RUNOFF'S PAGE MARGINS	6-2	RUNOFF COMMANDS, TYPES OF	5-4
REDEFINING RUNOFF'S PAGE SIZE	6-2	RUNOFF OUTPUT FILE	5-2
REMOTE	1-10	RUNOFF OUTPUT FILE	5-21
REMOTE/LOCAL	1-10	RUNOFF PROCESSING SEQUENCE, SUMMARY	5-19
REMOTE/LOCAL SWITCH	1-10	RUNOFF SOURCE FILE	5-2
RENAMING FILES AND SUB-UFDS	4-8	RUNOFF SPECIAL CHARACTERS	6-7
REPAIR PERSON	2-6	RUNOFF'S BLANK CHARACTER	6-7
REQUIRED BLANK CHARACTER IN RUNOFF	6-7	RUNOFF'S CHARACTER SYMBOL COMMANDS	5-5
REVISION BAR	9-18	RUNOFF'S COMMAND CONVENTIONS	5-4
RIGHT-ANGLE BRACKET	9-2	RUNOFF'S COMMAND FORMAT	9-1
RUBOUT	1-11	RUNOFF'S ERASE CHARACTER	6-7
		RUNOFF'S INDENTATION COMMANDS	6-3
		RUNOFF'S INDEXING COMMANDS	6-1

# MAIN INDEX TO THE NEW USER'S GUIDE

RUNOFF'S KILL CHARACTER	6-7	SOURCE FILE, PROCESSING	5-16
RUNOFF'S LINE-FORMATting COMMANDS	5-4	SOURCE FILE, SPECIFYING	5-18
RUNOFF'S LINE-FORMATting COMMANDS	5-7	SPECIAL CHARACTERS	1-10
RUNOFF'S MARGIN COMMANDS	6-2	SPECIAL CHARACTERS	1-11
RUNOFF'S MODES	5-7	SPECIAL CHARACTERS, RUNOFF	6-7
RUNOFF'S OUTPUT COMMANDS	5-4	SPECIAL CONVENTIONS, RUNOFF	6-9
RUNOFF'S OUTPUT COMMANDS	6-1	SPECIAL KEYS	1-10
RUNOFF'S OUTPUT OPTIONS	6-4	SPECIAL KEYS	1-11
RUNOFF'S PAGE SIZE COMMANDS	6-2	SPECIFYING INPUT FILE IN RUNOFF	5-18
RUNOFF'S PAGE-FORMATting COMMANDS	5-4	SPECIFYING SOURCE FILE IN RUNOFF	5-18
RUNOFF'S PAGE-FORMATting COMMANDS	6-1	SPOOL QUEUE	4-6
RUNOFF'S SPECIAL CONVENTIONS	6-9	SPOOL QUEUE	4-7
SAMPLE PRIMOS SESSION	2-9	SPOOL QUEUE LIST	4-6
SAMPLE SESSION,EDITOR	3-23	STARTING PAGE NUMBER	9-14
SAVING WORK FROM EDIT SESSION	3-19	STRING	1-8
SELECTING INPUT DEVICE FOR EDITOR	8-13	STRING	1-9
SEMICOLONS	3-4	STRING	3-2
SEMICOLONS FOLLOWING SEMICOLONS	3-5	STRING-FINDING COMMANDS, EDITOR'S	3-12
SEQUENTIAL PAGE NUMBER	9-24	SUB-UFD-NAME	4-2
SESSION	1-6	SUB-UFDS	4-2
SIDE MARGINS	6-2	SUB-UFDS, DELETING	4-8
SOURCE FILE	5-2	SUB-UFDS, EMPTY	4-9
		SUB-UFDS,RENAMING	4-8
		SUMMARY OF RUNOFF PROCESSING	
		SEQUENCE	5-19

# MAIN INDEX TO THE NEW USER'S GUIDE

SUPERVISOR	2-1	TERMINAL KEYBOARD	1-10
SUPPRESSING RUNOFF COMMAND ERROR MESSAGES	6-7	TERMINAL KEYBOARD LAYOUT	1-10
SUPPRESSING VERIFICATION OUTPUT	8-4	TERMINAL TIME	2-8
SWITCHING FROM EDIT TO INPUT MODE	3-8	TERMINAL, DISPLAYING RUNOFF OUTPUT ON	5-14
SWITCHING FROM INPUT TO EDIT MODE	3-5	TERMINAL, POWER SWITCH	2-3
SYMBOL VALUE	6-8	TERMINAL, REPAIR PERSON FOR	2-6
SYMBOL, RUNOFF, DEFINING	6-8	TERMINALS	1-9
SYMBOL-CHANGING COMMANDS, EDITOR'S	3-27	TERMINALS AND COMPUTERS	1-3
SYMBOL-NAME	6-8	TERMINALS, BATTERY	SEE UNDER HOOD
SYMBOLS, RUNOFF, RULES FOR DEFINING	6-9	TERMINALS, CATHODE RAY TUBE	1-9
TAB CHARACTER, DEFINING RUNOFF'S	5-9	TERMINALS, CPT	1-9
TAB CHARACTER, EDITOR'S	3-21	TERMINALS, HARD-COPY	1-9
TAB STOPS, EDITOR'S	3-21	TERMINALS, TYPES OF	1-9
TAB STOPS, PHYSICAL	8-27	TERMINALS, VIDEO	1-9
TABLE OF CONTENTS, MAKING, WITH RUNOFF	7-6	TERMINALS--SPECIAL CHARACTERS	1-11
TABS, DEFINING RUNOFF'S	5-9	TERMINALS--SPECIAL KEYS	1-11
TABS, EDITOR'S	3-21	TEXT	3-2
TABS, RUNOFF'S	5-9	TEXT COLUMNS, NUMBER OF	6-3
TELEPHONE	2-4	TEXT PARAMETERS	1-8
TELETYPE PAPER TYPE READER	8-13	TEXT PARAMETERS	3-2
TERMINAL CONTROLS AND SWITCHES	1-10	TEXT PERIOD, INITIAL	6-9
TERMINAL KEYBOARD	1-10	TEXT-PROCESSING SYSTEM	5-1
		TEXT-PROCESSOR	5-1
		THE ED COMMAND	3-3

PDR-3104

# MAIN INDEX TO THE NEW USER'S GUIDE

THE POINTER AND THE CURRENT LINE 3-6	USER FILE DIRECTORY NAME 2-1
TO INPUT A LINE FROM TERMINAL 2-2	USER-NAME 1-3
TOLLBOOTH, PHANTOM SEE LIBRARY	USING OTHER UFDS 4-1
TOP MARGIN 6-2	USING SLIST FOR SHORT FILES, 4-6
TRAILING BLANKS, CAN NOT APPEND 8-3	USING THE LINE PRINTER 4-5
TURNING TERMINAL ON 2-3	VALUE, RUNOFF SYMBOL 6-8
TYPES OF COMMANDS, RUNOFF'S 5-4	VALUE-SETTING COMMANDS, EDITOR'S 3-28
TYPES OF TERMINALS 1-9	VERIFICATION OUTPUT 8-33
TYPING ERRORS 2-2	VERIFICATION OUTPUT, SUPPRESSING 8-4
U/C 1-11	VIDEO TERMINALS 1-9
UFD 2-1	VITA BREVIS SEE APS LONGA
UFD 2-2	WHAT IS A SUB-UFD 4-2
UFD, CONTENTS OF 2-7	WHAT'S IN YOUR UFD 2-7
UFD-NAME 2-1	WHEN RUNOFF FINDS COMMANDS ERRORS 5-21
UFDS, USING OTHER 4-1	WIDTH OF PHYSICAL PAGE 9-25
UNDERLINING, IN RUNOFF 6-10	WORK FILE 3-3
UNRECOGNIZED COMMANDS 5-20	WORK SPACE 2-7
UP-ARROW 1-11	YOUR TERMINAL KEYBOARD 1-10
UPPER-CASE 1-11	1-11
UPPER-CASE/LOWER CASE 1-11	
UPPER-CASE/LOWER-CASE SWITCH 1-11	
UPPER/LOWER 1-11	
USER FILE DIRECTORY 2-1	
USER FILE DIRECTORY 2-2	

# PRIMOS COMMAND SUMMARY

<u>COMMAND</u>	<u>MEANING</u>
<u>ATTACH</u> { <u>ufd-name</u> sub-ufd-name <u>1/2</u> }	Make <u>ufd-name</u> or <u>sub-ufd-name</u> current UFD
<u>CNAME</u> old-filename new-filename	Change name of file <u>old-filename</u> to <u>new-filename</u>
<u>CREATE</u> sub-ufd-name	Make new sub-ufd <u>sub-ufd-name</u> in current UFD
<u>DELETE</u> { filename sub-ufd-name }	Delete <u>filename</u> or , if empty, <u>sub-ufd-name</u> from current UFD
<u>ED</u> [filename]	Invoke EDITOR to work on <u>filename</u> or to make a new file
<u>LISTF</u>	List contents of current UFD (names of files and sub-UFDs)
<u>LOGIN</u> ufd-name [password]	Log in to account <u>ufd-name</u> and ATTACH to <u>ufd-name</u> as current UFD
<u>LOGOUT</u>	Session is over--put away all files and free terminal for next user
<u>SLIST</u> filename	Display contents of <u>filename</u> at terminal
<u>RUNOFF</u> filename	RUNOFF the source file <u>filename</u>
<u>SPOOL</u> filename	Print out <u>filename</u> on Line Printer
<u>SPOOL</u> <u>-LIST</u>	List names of files in Spool Queue (waiting to be printed)
<u>SPOOL</u> <u>-CANCEL</u> <u>PRTxxx</u>	Remove file <u>PRTxxx</u> from Spool Queue

## EDITOR COMMAND SUMMARY

<u>COMMAND</u>	<u>MEANING</u>
<u>A</u> PPEND string	Append <u>string</u> to end of current line
<u>B</u> OTTOM	Move pointer to bottom of file
<u>B</u> RIEF	Suppress verification output
<u>C</u> HANGE/string-1/string-2/[G] [n]	Replace <u>string-1</u> with <u>string-2</u>
<u>D</u> ELETE [n]	Delete <u>n</u> lines
<u>D</u> ELETE <u>T</u> O string	Delete lines from work file until <u>string</u> found
<u>D</u> UNLOAD filename [n]	Copy <u>n</u> lines to <u>filename</u> & delete them from work file
<u>D</u> UNLOAD filename <u>T</u> O string	Copy lines to <u>filename</u> until <u>string</u> is found, then delete them from work file
<u>E</u> RASE character	Make <u>character</u> new Erase Character
<u>F</u> ILE [filename]	Turn work file into <u>filename</u>
<u>F</u> IND string	Find first line below current line beginning with <u>string</u>
<u>F</u> IND(n) string	Find first line below current line with <u>string</u> beginning in column <u>n</u>
<u>G</u> MODIFY	Alter current line as specified.
<u>I</u> NPOT { <u>(ASR)</u> <u>(PTR)</u> <u>(TFY)</u> }	Take input from this device
<u>I</u> NSERT newline	Insert <u>newline</u> below current line
<u>K</u> ILL character	Make <u>character</u> new Kill Character
<u>L</u> INESZ n	Set maximum line length to <u>n</u> characters (Default = 144)
<u>L</u> OAD filename	Copy contents of <u>filename</u> into work file just below current line
<u>L</u> OCATE string	Find first line below current line that contains <u>string</u> anywhere on that line

# EDITOR COMMAND SUMMARY

<u>COMMAND</u>	<u>MEANING</u>
<u>MODE</u> <u>COLUMN</u>	Display column number banner whenever INPUT Mode is entered.
<u>MODE</u> <u>NCOLUMN</u>	De-activate column banner (Default)
<u>MODE</u> <u>COUNT</u>	Activate counter symbol with indicated values
<u>MODE</u> <u>NCOUNT</u>	De-activate counter symbol
<u>MODE</u> <u>NUMBER</u>	Activate printing of line numbers
<u>MODE</u> <u>NNUMBER</u>	De-activate printing of line numbers (Default)
<u>MODE</u> <u>PROMPT</u>	Display prompts for INPUT and EDIT Modes.
<u>MODE</u> <u>NPROMPT</u>	De-activate Mode prompts (Default)
<u>MODE</u> $\left\{ \begin{array}{l} \underline{\text{PRALL}} \\ \underline{\text{PRUPPER}} \\ \underline{\text{PRLLOWER}} \end{array} \right\}$	Flag character cases on upper-case-only terminals.
<u>MODIFY</u> /string-1/string-2/ [G] [n]	Change <u>string-1</u> to <u>string-2</u> without altering character positions of other characters.
<u>MOVE</u> buffer-1 $\left\{ \begin{array}{l} \text{buffer-2} \\ \text{/string/} \end{array} \right\}$	Move <u>string</u> or contents of <u>buffer-2</u> into <u>buffer-1</u>
<u>NEXT</u> [n]	Move pointer <u>n</u> lines
<u>NFIND</u> string	Move pointer to first line below current line that does <u>not</u> being with <u>string</u>
<u>NFIND</u> (n) string	Move pointer to first line below current line that does <u>not</u> contain <u>string</u> in column <u>n</u>
<u>OUTPUT</u> $\left\{ \begin{array}{l} (\underline{\text{DISPLAY}}) \\ (\underline{\text{TTY}}) \end{array} \right\}$	Send verification output to indicated device (Default = TTY)
<u>OVERLAY</u> string	Superimpose <u>string</u> on current line
<u>PAUSE</u>	Freeze EDIT session & go to PRIMOS-level (Return via <u>START</u> )
<u>POINT</u> n	Move pointer to line <u>n</u>

# EDITOR COMMAND SUMMARY

<u>COMMAND</u>	<u>MEANING</u>
<u>PRINT</u> [n]	If <u>n</u> >0, print <u>n</u> lines. If <u>n</u> <0, back up <u>n</u> lines and print 1 line.
<u>PSYMBOL</u>	Print list of current Reserved Characters
<u>PTABSET</u> ptab-1 ptab2 ... ptab-8	Indicate location of physical physical tab stop settings on current output device
<u>PUNCH</u> $\left\{ \begin{array}{l} \text{(ASR)} \\ \text{(PTP)} \end{array} \right\}$ n	Punch <u>n</u> lines on indicated device
<u>QUIT</u>	Leave EDITOR without filing results of session--retains original file unchanged
<u>RETYPE</u> string	Delete current line and replace it with <u>string</u>
<u>SYMBOL</u> name character	Change current value of <u>character</u> to <u>name</u>
<u>TABSET</u> tab-2 tab-2...tab-8	Set these logical tab stops for use when EDITOR tab symbol is inputted
<u>TOP</u>	Move pointer to top of work file
<u>UNLOAD</u> filename [n]	Copy <u>n</u> lines from work file into <u>filename</u>
<u>UNLOAD</u> filename <u>TO</u> string	Copy lines from work file into <u>filename</u> until <u>string</u> is found
<u>VERFIY</u>	Print verification lines (Default)
<u>WHERE</u>	Print current line number
<u>XEQ</u> buffer	Execute contents of <u>buffer</u> as a command line
<u>*</u> [n]	Repeat commands preceding <u>*</u> either <u>n</u> times or until bottom of file is hit.



# RUNOFF COMMAND SUMMARY

<u>COMMAND</u>	<u>IMPLICIT</u>		<u>MEANING</u>
	<u>BRK</u>	<u>EJT</u>	
	X = Yes		
	? = Conditional		
.			Start processing
.*text			Comment line--will not appear in output file
.+ text	X		Enter <u>text</u> verbatim--exactly as shown
.> text	X		Center <u>text</u> on line
./left/center/right/	X		Apportion text into left-justified, centered and right-justified portions
. <u>ADJUST</u>	X		<u>Adjust</u> filled output lines (Default)
. <u>BLANK</u> character			Make <u>character</u> new Required Blank
. <u>BMARGIN</u> n	X	X	Set bottom margin to <u>n</u> lines(Default 5)
. <u>BREAK</u>			Do not fill or adjust most recent output line
. <u>CMARGIN</u> m	X	X	Set inter-column margin to <u>m</u> spaces (Default 5)
. <u>COLUMNS</u> i	X	X	Make <u>i</u> columns of text (Default 1)
. <u>DEFINE</u> symbol-name value			Replace all coming occurrences of in source file with <u>value</u> in output file
. <u>EFooter</u> /left/center/right			Define footer for upcoming even-numbered pages
. <u>EHeader</u> /left/center/right			Define header for upcoming even-numbered pages
. <u>EJECT</u>	X	X	Proceed to top of next printing page
. <u>ERASE</u> character			Make <u>character</u> new Erase Character
. <u>ERRGO</u>			Suppress Error Messages & Prompts during processing
. <u>FILE</u> filename			File output in <u>filename</u>
. <u>FILL</u>	X		<u>Fill</u> but do <u>not</u> adjust text

# RUNOFF COMMAND SUMMARY

<u>COMMAND</u>	<u>IMPLICIT</u>		<u>BRK</u>	<u>EJT</u>	<u>MEANING</u>
					X = Yes ? = Conditional
<u>.FLOAT</u> filename					Insert floating file
<u>.FOOTER</u> /left/center/right/					Put this footer on all subsequent pages
<u>.FROM</u> i					Display/file output beginning on page <u>i</u>
<u>.HEADER</u> /left/center/right/					Put this header on all subsequent pages
<u>.HYPHEN</u> character					Make <u>character</u> new Phantom Hyphen
<u>.INDENT</u> [m]					Indent left margin by <u>m</u> more spaces (Default 5)
<u>.INDEX</u> string					Enter <u>string</u> and current page number into Index File
<u>.INSERT</u> filename [0,1...9]					Insert <u>filename</u>
<u>.IXFILE</u> filename					Copy entries flagged by <u>.INDEX</u> into <u>filename</u>
<u>.KILL</u> character					Make <u>character</u> new Kill Character
<u>.LENGTH</u> n	X	X			Define length of physical page as <u>n</u> lines. (Default = 66 lines = 11 inches)
<u>.NADJUST</u>					Stop <u>adjusting</u> output lines.
<u>.NEED</u> [n]			?	?	Requires unbroken block of <u>n</u> lines on currentpage at once. The BREAK & EJECT are done only if these line are <u>not</u> available on current line
<u>.NERRGO</u>					Stop processing source file & display error message when command errors are found (Default mode)
<u>.NFILE</u>					Do not put output into a file.
<u>.NFILL</u>			X		Do <u>not</u> FILL or ADJUST text. Recognize tab character anywhere on input lines
<u>.NIXFILE</u>					Do <u>not</u> generate index file--Ignore all INDEX commands

# RUNOFF COMMAND SUMMARY

<u>COMMAND</u>	<u>IMPLICIT</u>		<u>BRK</u>	<u>EJT</u>	<u>MEANING</u>
					X = Yes ? -- Conditional
<u>.NPARAGRAPH</u>					Reset paragraph values to Indent 0, Skip 1
<u>.NPAUSE</u>					Do <u>not</u> pause between output pages (Default)
<u>.NPERFORATE</u>					Do <u>not</u> print perforation marks between pages
<u>.NTTY</u>					Do not display output at the terminal (Default)
<u>.OFOOTER</u> /left/center/right/					Define footer on odd-numbered pages
<u>.OHEADER</u> /left/center/right/					Define header on odd-numbered pages
<u>.PAGEN</u> i					Define new page number (replaces "#" in header&footer formats)
<u>.PARAGRAPH</u> [m] [n]			X		Begin new paragraph--Indent <u>m</u> spaces, skip <u>n</u> lines (Default 0 spaces, 1 line)
<u>.PAUSE</u>					Pause between each output page
<u>.PERFORATE</u> [n]					Print perforation marks between pages
<u>.PICTURE</u> n			?	?	Reserve <u>n</u> physical lines for later insertion of artwork
<u>.PURGE</u>					Force immediate satisfaction of all outstanding PICTURE and FLOAT files& commands
<u>.QUIT</u>			X	X	Return to previous level of authority
<u>.RBAR</u> [ON]					Turns on Revision Bar
<u>.RETURN</u> i					Returns you to previous source file
<u>.RINDENT</u> [m]					Increase right margin by <u>m</u> spaces (Default 5)
<u>.RUNDENT</u> [m]					Decrease right margin by <u>m</u> spaces (Default = original margin)

# RUNOFF COMMAND SUMMARY

<u>COMMAND</u>	<u>IMPLICIT</u>		<u>MEANING</u>
	<u>BRK</u>	<u>EJT</u>	
	Y = Yes		
	? = Conditional		
<u>.SKIP</u> {n}	X		Skip <u>n</u> printing lines
<u>.SMARGIN</u> [m]	X	X	Reset side margins to <u>m</u> spaces each (Default = 7 spaces)
<u>.SOURCE</u> [n]			Print source file line numbers on output file
<u>.SPACE</u> [n]			Set spacing to <u>n</u> (Default = 1 = Single- spacing)
<u>.STOP</u>	?	?	Conditional QUIT
<u>.SYCHAR</u> character			Define symbol-name delimiter
<u>.TAB</u> character tab-1 tab-2 ... tab-20			Define Tab Character and tab stops (No default values)
<u>.TMARGIN</u> [n]	X	X	Redefine top margin (Default = 7 lines)
<u>.TO</u> i			Define page number of last processed page wanted
<u>.TTY</u>			Display processed output at terminal
<u>.UNDEFINE</u> [symbol]			Undefine specified <u>symbol</u> or all symbols
<u>UNDENT</u> [m]			Move left margin <u>m</u> spaces to left (Default = original margin)
<u>.WIDOW</u> [n]			Avoid widows of <u>n</u> lines by EJECTing
<u>.WIDTH</u> [m]	X	X	Redefine physical page width, to <u>m</u> spaces, including margins (Default = 85 10-pitch spaces = 8 1/2 inches)

# INDEX OF SYSTEM MESSAGES

*****ERROR(S)*****	5-21	RESERVED CHARACTER	2-5
? 1-7		UNRECOGNIZED	5-21
? 3-9			
ALREADY EXISTS	4-7		
BAD	3-9		
BAD PASSWORD(ATCH\$\$)	2-6		
BAD PRINT FILE NAME	4-7		
CAN'T CANCEL REQUEST	4-7		
DIRECTORY NOT EMPTY	4-8		
ENTER OUTPUT FILE TREENAME:			
5-19			
ER1	1-7		
FILE MODIFIED, OK TO QUIT?			
3-19,20			
FILE NAME MUST BE SPECIFIED			
3-20			
ILL REM REF.	2-5		
ILLEGAL COMMAND	5-20		
INPUT FILE:	5-18		
LOGIN PLEASE	2-5		
NO RIGHT(LOGIN)	2-5		
NOT A UFD	2-6		
NOT A UFO3-7			
OK TO DELETE OLD FILENAME ?			
5-19, 6-5			
PLEASE SPECIFY NEW FILENAME:			
5-19			
PROCESSING...	5-18		
PRTXXX NOT IN QUEUE	4-7		